User's guide for iREG (Identification and Controller Design and Reduction)

Ioan Doré Landau and Tudor-Bogdan Airimiţoaie

December 14, 2013

Contents

Chapter 1

About iREG

iREG (Identification and Robust Digital Controller Design) is a software product designed in Visual C++. It is a user friendly application with graphical user interface (GUI) for identifying SISO (Single Input Single Output) processes and designing robust digital controllers for SISO plants. There are various routines for identification in opened and closed loop. The controller design procedure is the combined pole placement with sensitivity functions shaping. The theoretical background behind these procedures can be found in [2] and [5]. The program was developed in GIPSA-LAB, Grenoble, by T.B. Airimitoaie under the supervision of Professor I.D. Landau. It has commenced as an interactive software for the design of robust digital controllers during a final year project "T.B. Airimitoaie, Interactive C program for the design of robust digital controllers" and continued with the addition of open loop and closed loop identification routines during the PhD thesis of T.B. Airimitoaie.

1.1 Software requirements

The iREG program was developed under Visual C++ environment and needs one of Microsofts Operating Systems to work properly. The software can read or write Matlab version 6 files (*.mat) without an existing installation of the Matlab software.

1.2 Getting started

- To run iREG one has to double click the icon entitled iREG.exe. A window like the one in Fig.1.1 will appear.
- Next you should select the proper button depending on what you want to do (identify a plant model, design a controller (robust digital or continuous) or compute a reduced order controller).
- The program offers the possibility to switch between English and French interface.

1.3 Description of the software

After double clicking the program icon, 2 windows will be displayed: the main window (Figure 1.1) witch gives access to all the functionality of the program and the plots window witch shows the graphics. Each window is organized in tabs. The main window contains the following tabs:

- *Start* - starting tab of the application, the different functionalities of the program (identification, controller design or controller reduction) can be activated from this tab; furthermore, it an be used to save/load the workspace;
- *Identification* - this tab gives access to all identification routines implemented in iREG and can also be used to modify the graphics associated with the identification part of this software;

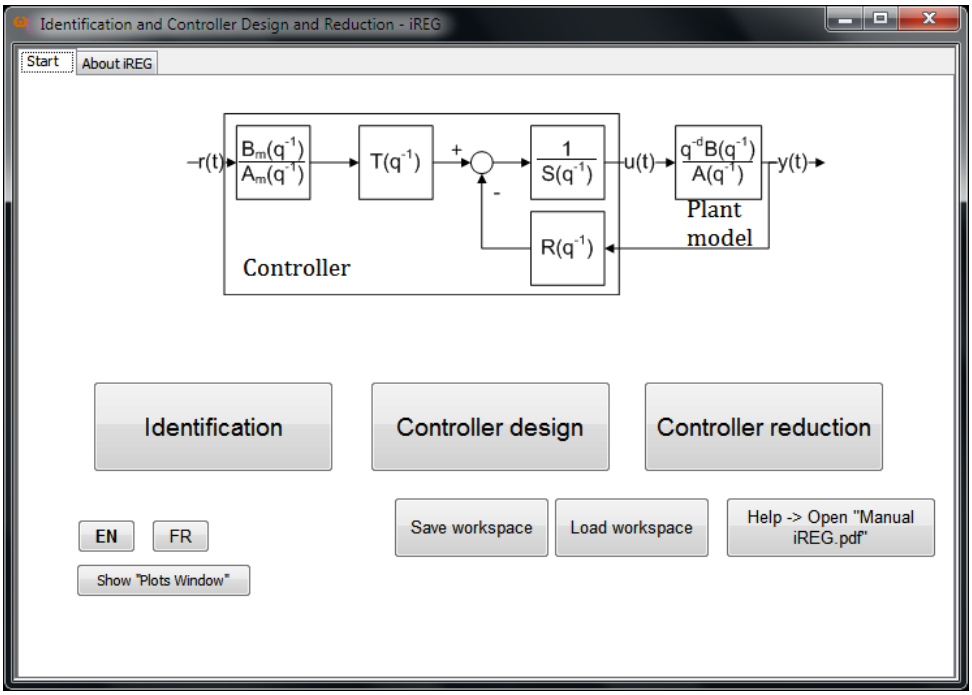


Figure 1.1: Start window of the application

- *Architecture* - includes all necessary functionality represents the initial tab for designen a controller; one can select the type of controller to be design (digital RST or continuous PI/PID), load the plant model, save the controller or test an already computed controller;
- *Controller Design* - offers all necessary tools to design and properly adjust all parameters of a robust digital controller;
- *Automated controller Design* - offers only the basic tools to design and adjust a robust digital controller;
- *PI/PID* - offers the possibility of implementing a continuous PI or PID controller;
- *Analysis Plots* - can be used to display graphics and modify their appearance;
- *Band-stop Filters* - used to implement filters for the precise shaping of the sensibility functions;
- *Tracking* - for designing the tracking part of a digital controller (reference model and polynomial T);
- *Reduc* - can be used to reduce the order of a RST controller;
- *About iREG* - information regarding the software (name, version, authors and copyright).

Part I

Identification

Chapter 2

Basic theory

The recursive parameter estimation principle for sampled models is illustrated in Figure 2.1.

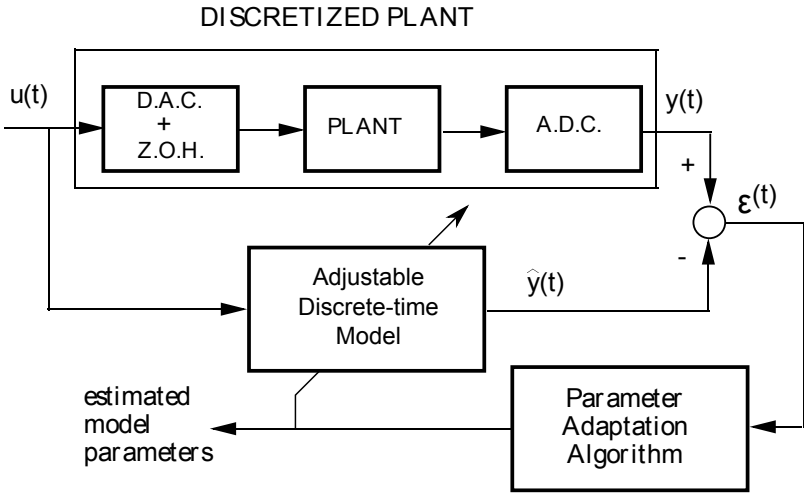


Figure 2.1: Parameter estimation principle

A discrete time model with adjustable parameters is implemented on the computer. The error between the system output at instant t, $y(t)$ and the output predicted by the model $\hat{y}(t)$ (called plant-model error or prediction error) is used by the *parameter adaptation algorithm*, which, at each sampling instant, will modify the model parameters in order to minimize this error (in the sense of a certain criterion).
The input to the system is a low level and frequency rich signal generated by the computer, for the case of plant model identification in open loop, or a combination of this and the signal generated by the controller in the case of closed loop identification.
The key element for implementing the on-line estimation of the plant model parameters is the *parameter adaptation algorithm* (PAA) which drives the parameters of the adjustable prediction model from the data acquired on the system at each sampling instant. This algorithm has a *recursive* structure, i.e., the new value of the estimated parameters is equal to the previous value plus a correcting term which will depend on the most recent measurements.
In general a *parameter vector* is defined. Its components are the different parameters that should be estimated.

The parameter adaptation algorithms generally have the following structure:

$$\begin{bmatrix} New\ estimated \\ parameters \\ (vector) \end{bmatrix} = \begin{bmatrix} Previous\ estimated \\ parameters \\ (vector) \end{bmatrix} + \begin{bmatrix} Adaptation \\ gain \\ (matrix) \end{bmatrix} \times \begin{bmatrix} Measurement \\ function \\ (vector) \end{bmatrix} \times \begin{bmatrix} Prediction\ error \\ function \\ (scalar) \end{bmatrix}$$

This structure corresponds to the so-called *integral type adaptation algorithms* (the algorithm has memory and therefore maintains the estimated value of the parameters when the correcting terms become null). The algorithm can be viewed as a discrete-time integrator fed at each instant by the correcting term. The measurement function vector is generally called the *observation vector*. The prediction error function is generally called the *adaptation error*.
As will be shown, there are more general structures where the integrator is replaced by other types of dynamics, i.e., the new value of the estimated parameters will be equal to a function of the previous parameter estimates (eventually over a certain horizon) plus the correcting term.
The adaptation gain plays an important role in the performances of the parameter adaptation algorithm and it may be constant or time-varying.
The problem addressed in this chapter is the synthesis and analysis of parameter adaptation algorithms in a deterministic environment.

2.1 Gradient Algorithm

The aim of the gradient parameter adaptation algorithm is to minimize a quadratic criterion in terms of the prediction error.
Consider the discrete-time model of a plant described by:

$$y(t+1) = -a_1 y(t) + b_1 u(t) = \theta^T \phi(t) \tag{2.1}$$

where the unknown parameters a_1 and b_1 form the components of the *parameter vector* θ:

$$\theta^T = [a_1, b_1] \tag{2.2}$$

and

$$\phi^T(t) = [-y(t), u(t)] \tag{2.3}$$

is the *measurement vector*.
The adjustable prediction model will be described in this case by:

$$\hat{y}^0(t+1) = \hat{y}[(t+1)/\hat{\theta}(t)] = -\hat{a_1}(t)y(t) + \hat{b}(t)u(t) = \hat{\theta^T}(t)\phi(t) \tag{2.4}$$

where $\hat{y}^0(t+1)$ is termed the *a priori* predicted output depending on the value of the *estimated parameter vector* at instant t,

$$\hat{\theta^T}(t) = [\hat{a_1}(t), \hat{b_1}(t)] \tag{2.5}$$

As it will be shown later, it is very useful to consider also the *a posteriori* predicted output computed on the basis of the new estimated parameter vector at $t+1$, $\hat{\theta}(t+1)$, which will be available somewhere between $t+1$ and $t+2$. The *a posteriori* predicted output will be given by:

$$\begin{aligned} \hat{y}(t+1) &= \hat{y}[(t+1)|\hat{\theta}(t+1)] \\ &= -\hat{a}_1(t+1)y(t) + \hat{b}_1(t+1)u(t) \\ &= \hat{\theta}^T(t+1)\phi(t) \end{aligned} \tag{2.6}$$

One defines an *a priori* prediction error,

$$\epsilon^0(t+1) = y(t+1) - \hat{y}^0(t+1) \tag{2.7}$$

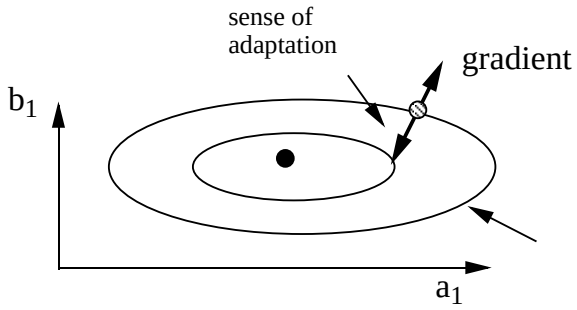


Figure 2.2: Principle of the gradient method

and an *a posteriori* prediction error,

$$\epsilon(t+1) = y(t+1) - \hat{y}(t+1) \tag{2.8}$$

The objective is to find a recursive parameter adaptation algorithm with memory. The structure of such an algorithm is:

$$\hat{\theta}(t+1) = \hat{\theta}(t) + \Delta\hat{\theta}(t+1) = \hat{\theta}(t) + f[\hat{\theta}(t), \phi(t), \epsilon^0(t+1)] \tag{2.9}$$

The correction term $f[\hat{\theta}(t), \phi(t), \epsilon^0(t+1)]$ must depend solely on the information available at the instant $(t+1)$ when $y(t+1)$ is acquired (last measurement $y(t+1)$, $\hat{\theta}(t)$, and a finite amount of information at times t, $t-1$, $t-2, \cdots, t-n$). The correction term must enable the following criterion to be minimized at each step:

$$\min_{\hat{\theta}(t)} J(t+1) = [\epsilon^0(t+1)]^2 \tag{2.10}$$

A solution can be provided by the gradient technique.
If the iso-criterion curves ($J = constant$) are represented in the plane of the parameters a_1, b_1, concentric closed curves are obtained around the minimum value of the criterion, which is reduced to the point (a_1, b_1) corresponding to the parameters of the plant model. As the value of $J = const.$ increases, the iso-criterion curves move further and further away from the minimum. This is illustrated in Figure 2.2. In order to minimize the value of the criterion, one moves in the opposite direction of the gradient to the corresponding iso-criterion curve. This will lead to a curve corresponding to $J = const$, of a lesser value, as is shown in Figure 2.2. The corresponding parametric adaptation algorithm will have the form:

$$\hat{\theta}(t+1) = \hat{\theta}(t) - F\frac{\delta J(t+1)}{\delta\hat{\theta}(t)} \tag{2.11}$$

where $F = \alpha I(\alpha > 0)$ is the matrix adaptation gain (I - unitary diagonal matrix) and $\delta J(t+1)/\delta\hat{\theta}(t)$ is the gradient of the criterion given in (2.10) with respect to $\hat{\theta}(t)$. From (2.10) one obtains:

$$\frac{1}{2}\frac{\delta J(t+1)}{\delta\hat{\theta}(t)} = \frac{\delta\epsilon^0(t+1)}{\delta\hat{\theta}(t)}\epsilon^0(t+1) \tag{2.12}$$

But:

$$\epsilon^0(t+1) = y(t+1) - \hat{y^0}(t+1) = y(t+1) - \hat{\theta}^T(t)\phi(t) \tag{2.13}$$

and

$$\frac{\delta\epsilon^0(t+1)}{\delta\hat{\theta}(t)} = -\phi(t) \tag{2.14}$$

Introducing (2.14) in (2.12), the parameter adaptation algorithm of (2.11) becomes:

$$\boxed{\hat{\theta}(t+1) = \hat{\theta}(t) + F\phi(t)\epsilon^0(t+1)} \tag{2.15}$$

where F is the matrix adaptation gain. There are two possible choices:

1. $F = \alpha I$; $\alpha > 0$
2. $F > 0$ (positive definite matrix)[1]

The resulting algorithm has an integral structure. Therefore it has memory (for $\varepsilon^0(t+1) = 0$, $\hat{\theta}(t+1) = \hat{\theta}(t)$). The geometrical interpretation of the PAA of (2.15) is given in Figure 2.3.
The correction is done in the direction of the observation vector (which in this case is the measurement vector) in the case $F = \alpha I$, $\alpha > 0$ or within ± 90 deg around this direction when $F > 0$ (a positive definite matrix may cause a rotation of a vector for less than 90 deg).

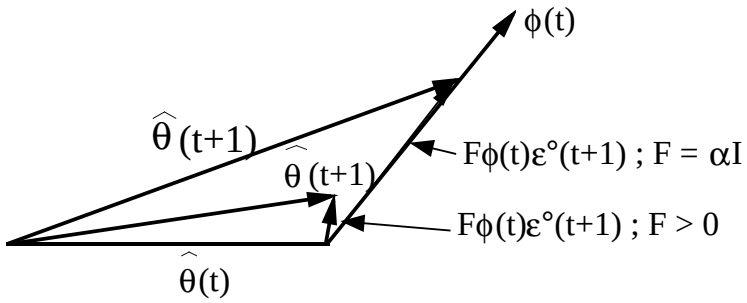


Figure 2.3: Geometrical interpretation of the gradient adaptation algorithm

The parameter adaptation algorithm given in (2.15) presents instability risks if a large adaptation gain (respectively a large α) is used. This can be understood by referring to Figure 2.2. If the adaptation gain is large near the optimum, one can move away from this minimum instead of getting closer.
The following analysis will allow to establish necessary conditions upon the adaptation gain in order to avoid instability.
Consider the parameter error defined as:

$$\tilde{\theta}(t) = \hat{\theta}(t) - \theta \tag{2.16}$$

From Eqs. 2.1 and 2.4 it results:

$$\epsilon^0(t+1) = y(t+1) - \hat{y}^0(t+1) = \theta^T \phi(t) - \hat{\theta}^T(t)\phi(t) = -\phi^T(t)\tilde{\theta}(t) \tag{2.17}$$

Subtracting θ in the two terms of (2.15) and using (2.17) one gets:

$$\begin{aligned} \tilde{\theta}(t+1) &= \tilde{\theta}(t) - F(t)\phi(t)\phi^T(t)\tilde{\theta}(t) = [I - F\phi(t)\phi^T(t)]\tilde{\theta}(t) \\ &= A(t)\tilde{\theta}(t) \end{aligned} \tag{2.18}$$

(2.18) corresponds to a time-varying dynamical system. A necessary stability condition (but not sufficient) is that the eigen values of $A(t)$ be inside the unit circle at each instant t. This leads to the following condition for the choice of the adaptation gain as $F = \alpha I$:

$$\alpha < \frac{1}{\phi^T(t)\phi(t)} \tag{2.19}$$

2.1.1 Improved Gradient Algorithm

In order to assure the stability of the PAA for any value of the adaptation gain α (or of the eigenvalues of the gain matrix F) the same gradient approach is used but a different criterion is considered:

$$\min_{\hat{\theta}(t+1)} J(t+1) = [\epsilon(t+1)]^2 \tag{2.20}$$

[1] A symmetric square matrix F is termed positive definite if $x^T F x > 0$ for all $x \neq 0$, $x \in \Re^n$. In addition: (i) all the terms of the main diagonal are positive, (ii) the determinants of all the principals minors are positive.

The equation:

$$\frac{1}{2}\frac{\delta J(t+1)}{\delta\hat{\theta}(t+1)} = \frac{\delta\epsilon(t+1)}{\delta\hat{\theta}(t+1)}\epsilon(t+1) \tag{2.21}$$

is then obtained. From (2.6) and (2.8) it results that:

$$\epsilon(t+1) = y(t+1) - \hat{y}(t+1) = y(t+1) - \hat{\theta}^T(t+1)\phi(t) \tag{2.22}$$

and, respectively:

$$\frac{\delta\epsilon(t+1)}{\delta\hat{\theta}(t+1)} = -\phi(t) \tag{2.23}$$

Introducing (2.23) in (2.21), the parameter adaptation algorithm of (2.11) becomes:

$$\hat{\theta}(t+1) = \hat{\theta}(t) + F\phi(t)\epsilon(t+1) \tag{2.24}$$

This algorithm depends on $\varepsilon(t+1)$, which is a function of $\hat{\theta}(t+1)$. For implementing this algorithm, $\varepsilon(t+1)$ must be expressed as a function of $\varepsilon^0(t+1)$, i.e. $\varepsilon(t+1) = f[\hat{\theta}(t), \phi(t), \varepsilon^0(t+1)]$
(2.22) can be rewritten as:

$$\epsilon(t+1) = y(t+1) - \hat{\theta}^T(t)\phi(t) - [(\hat{\theta}(t+1) - \hat{\theta}(t)]^T\phi(t) \tag{2.25}$$

The first two terms of the right hand side correspond to $\varepsilon^0(t+1)$, and from (2.24), one obtains:

$$\hat{\theta}(t+1) - \hat{\theta}(t) = F\phi(t)\epsilon(t+1) \tag{2.26}$$

which enables to rewrite (2.25) as:

$$\epsilon(t+1) = \epsilon^0(t+1) - \phi^T(t)F\phi(t)\epsilon(t+1) \tag{2.27}$$

from which the desired relation between $\varepsilon(t+1)$ and $\varepsilon^0(t+1)$ is obtained:

$$\epsilon(t+1) = \frac{\epsilon^0(t+1)}{1+\phi^T(t)F\phi(t)} \tag{2.28}$$

and the algorithm of (2.24) becomes:

$$\hat{\theta}(t+1) = \hat{\theta}(t) + \frac{F\phi(t)\epsilon^0(t+1)}{1+\phi^T(t)F\phi(t)} \tag{2.29}$$

which is a *stable algorithm* irrespective of the value of the gain matrix F (positive definite).
The division by $1 + \phi^T(t)F\phi(t)$ introduces a normalization with respect to F and $\phi(t)$ which reduces the sensitivity of the algorithm with respect to F and $\phi(t)$.
In this case, the equation for the evolution of the parametric error $\hat{\theta}(t)$ is given by:

$$\tilde{\theta}(t+1) = \left[I - \frac{F\phi(t)\phi^T(t)}{1+\phi^T(t)F\phi(t)}\right]\tilde{\theta}(t) = A(t)\tilde{\theta}^T(t) \tag{2.30}$$

and the eigenvalues of $A(t)$ will always be inside or on the unit circle, but this is not enough to conclude upon the stability of the algorithm.

2.2 Recursive Least Squares Algorithm

When using the Gradient Algorithm, $\epsilon^2(t+1)$ is minimized at each step or, to be more precise, one moves in the quickest decreasing direction of the criterion, with a step depending on F. The minimization of $\epsilon^2(t+1)$ at

each step does not necessarily lead to the minimization of:

$$\sum_{i=1}^{t} \epsilon^2(i+1)$$

on a time horizon, as is illustrated in Figure 2.4. In fact, in the vicinity of the optimum, if the gain is not low enough, oscillations may occur around the minimum. On the other hand, in order to obtain a satisfactory convergence speed at the beginning when the optimum is far away, a high adaptation gain is preferable. In fact, the least squares algorithm offers such a variation profile for the adaptation gain. The same equations as in the gradient algorithm are considered for the plant, the prediction model, and the prediction errors, namely (2.1) through (2.8).
The aim is to find a recursive algorithm of the form of (2.9) which minimizes the *least squares* criterion:

$$\min_{\hat{\theta}(t)} J(t) = \sum_{i=1}^{t} [y(i) - \hat{\theta}^T(t)\phi(i-1)]^2 \tag{2.31}$$

The term $\hat{\theta}(t)^T\phi(i-1)$ corresponds to:

$$\hat{\theta}^T(t)\phi(i-1) = -\hat{a}_1(t)y(i-1) + \hat{b}_1(t)u(i-1) = \hat{y}[i \mid \hat{\theta}(t)] \tag{2.32}$$

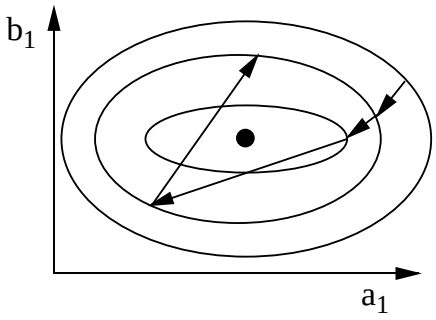


Figure 2.4: Evolution of an adaptation algorithm of the gradient type

Therefore, this is the prediction of the output at instant $i(i \leq t)$ based on the parameter estimate at instant t obtained using t measurements.
First, a parameter θ must be estimated at instant t so that it minimizes the sum of the squares of the differences between the output of the plant and the output of the prediction model on a horizon of t measurements. The value of $\hat{\theta}(t)$, which minimizes the criterion (2.31), is obtained by seeking the value that cancels $\delta J(t)/\delta\hat{\theta}(t)$:

$$\frac{\delta J(t)}{\delta\hat{\theta}(t)} = -2\sum_{i=1}^{t} [y(i) - \hat{\theta}^T(t)\phi(i-1)]\phi(i-1) = 0 \tag{2.33}$$

From (2.33), taking into account that:

$$[\hat{\theta}^T(t)\phi(i-1)]\phi(i-1) = \phi(i-1)\phi^T(i-1)\hat{\theta}(t)$$

one obtains:

$$\left[\sum_{i=1}^{t} \phi(i-1)]\phi^T(i-1)\right] \hat{\theta}(t) = \sum_{i=1}^{t} y(i)\phi(i-1)$$

and left multiplying by[2]:

$$\left[\sum_{i=1}^{t} \phi(i-1)\phi^T(i-1)\right]^{-1}$$

[2]It is assumed that the matrix $\sum_{i=1}^{t} \phi(i-1)\phi^T(i-1)$ is invertible. As it will be shown later this corresponds to an *excitation* condition

one obtains:

$$\begin{aligned}\hat{\theta}(t) &= \left[\sum_{i=1}^{t}\phi(i-1)\phi^T(i-1)\right]^{-1}\sum_{i=1}^{t}y(i)\phi(i-1)\\ &= F(t)\sum_{i=1}^{t}y(i)\phi(i-1)\end{aligned} \tag{2.34}$$

in which:

$$F(t)^{-1} = \sum_{i=1}^{t}\phi(i-1)\phi^T(i-1) \tag{2.35}$$

This estimation algorithm is not recursive. In order to obtain a recursive algorithm, the estimation of $\hat{\theta}(t+1)$ is considered:

$$\hat{\theta}(t+1) = F(t+1)\sum_{i=1}^{t+1}y(i)\phi(i-1) \tag{2.36}$$

$$F(t+1)^{-1} = \sum_{i=1}^{t+1}\phi(i-1)\phi^T(i-1) = F(t)^{-1} + \phi(t)\phi^T(t) \tag{2.37}$$

We can now express $\hat{\theta}(t+1)$ as a function of $\hat{\theta}(t)$:

$$\hat{\theta}(t+1) = \hat{\theta}(t) + \Delta\hat{\theta}(t+1) \tag{2.38}$$

From (2.37) one has:

$$\hat{\theta}(t+1) = F(t+1)\left[\sum_{i=1}^{t}y(i)\phi(i-1) + y(t+1)\phi(t)\right] \tag{2.39}$$

Taking into account (2.34), (2.39) can be rewritten as:

$$\hat{\theta}(t+1) = F(t+1)[F(t)^{-1}\hat{\theta}(t) + y(t+1)\phi(t)] \tag{2.40}$$

From (2.37) after post-multiplying both sides by $\hat{\theta}(t)$ one gets:

$$F(t)^{-1}\hat{\theta}(t) = F(t+1)^{-1}\hat{\theta}(t) - \phi(t)\phi^T(t)\hat{\theta}(t) \tag{2.41}$$

and (2.40), becomes:

$$\hat{\theta}(t+1) = F(t+1)\left\{F(t+1)^{-1}\hat{\theta}(t) + \phi(t)[y(t+1) - \hat{\theta}^T(t)\phi(t)]\right\} \tag{2.42}$$

Taking into account the expression of $\epsilon^0(t+1)$ given by (2.13), the result is:

$$\hat{\theta}(t+1) = \hat{\theta}(t) + F(t+1)\phi(t)\epsilon^0(t+1) \tag{2.43}$$

The adaptation algorithm of (2.43) has a recursive form similar to the gradient algorithm given in (2.15) except that the gain matrix $F(t+1)$ is now time-varying since it depends on the measurements (it automatically corrects the gradient direction and the step length). A recursive formula for $F(t+1)$ remains to be given from the recursive formula $F^{-1}(t+1)$ given in (2.37). This is obtained by using the *matrix inversion* lemma.

Matrix Inversion Lemma: *Let F be a $(n \times n)$ dimensional nonsingular matrix, R a $(m \times m)$ dimensional nonsingular matrix and H a $(n \times m)$ dimensional matrix of maximum rank, then the following identity holds:*

$$(F^{-1} + HR^{-1}H^T)^{-1} = F - FH(R + H^TFH)^{-1}H^TF \tag{2.44}$$

Proof: By direct multiplication one finds that:

$$[F - FH(R + H^T FH)^{-1} H^T F][F^{-1} + HR^{-1} H^T] = I$$

For the case of (2.37), one chooses $H = \phi(t)$, $R = 1$ and one obtains from (2.37) and (2.44):

$$F(t+1) = F(t) - \frac{F(t)\phi(t)\phi^T(t)F(t)}{1 + \phi^T(t)F(t)\phi(t)} \tag{2.45}$$

and, putting together the different equations, a first formulation of the recursive least squares (RLS) parameter adaptation algorithm (PAA) is given below:

$$\hat{\theta}(t+1) = \hat{\theta}(t) + F(t+1)\phi(t)\epsilon^0(t+1) \tag{2.46}$$

$$F(t+1) = F(t) - \frac{F(t)\phi(t)\phi^T(t)F(t)}{1 + \phi^T(t)F(t)\phi(t)} \tag{2.47}$$

$$\epsilon^0(t+1) = y(t+1) - \hat{\theta}^T(t)\phi(t) \tag{2.48}$$

An equivalent form of this algorithm is obtained by introducing the expression of $F(t+1)$ given by (2.47) in (2.46), where:

$$\begin{aligned} [\hat{\theta}(t+1) - \hat{\theta}(t)] &= F(t+1)\phi(t)\epsilon^0(t+1) \\ &= F(t)\phi(t)\frac{\epsilon^0(t+1)}{1 + \phi^T(t)F(t)\phi(t)} \end{aligned} \tag{2.49}$$

However, from (2.7), (2.8) and (2.49), one obtains:

$$\begin{aligned} \epsilon(t+1) = y(t+1) - \hat{\theta}^T(t+1)\phi(t) &= y(t+1) - \hat{\theta}(t)\phi(t) \\ -[\hat{\theta}(t+1) - \hat{\theta}(t)]^T\phi(t) &= \epsilon^0(t+1) \\ -\phi^T(t)F(t)\phi(t)\frac{\epsilon^0(t+1)}{1 + \phi^T(t)F(t)\phi(t)} &= \frac{\epsilon^0(t+1)}{1 + \phi^T(t)F(t)\phi(t)} \end{aligned} \tag{2.50}$$

which expresses the relation between the *a posteriori* prediction error and the *a priori* prediction error. Using this relation in (2.49), an equivalent form of the parameter adaptation algorithm for the recursive least squares is obtained:

$$\hat{\theta}(t+1) = \hat{\theta}(t) + F(t)\phi(t)\epsilon(t+1) \tag{2.51}$$

$$F(t+1)^{-1} = F(t)^{-1} + \phi(t)\phi^T(t) \tag{2.52}$$

$$F(t+1) = F(t) - \frac{F(t)\phi(t)\phi^T(t)F(t)}{1 + \phi^T(t)F(t)\phi(t)} \tag{2.53}$$

$$\epsilon(t+1) = \frac{y(t+1) - \hat{\theta}^T(t)\phi(t)}{1 + \phi^T(t)F(t)\phi(t)} \tag{2.54}$$

For the recursive least squares algorithm to be exactly equivalent to the non-recursive least squares algorithm, it must be started from a first estimation obtained at instant $t_0 = \dim \phi(t)$, since normally $F(t)^{-1}$ given by (2.35) becomes nonsingular for $t = t_0$. In practice, the algorithm is started up at $t = 0$ by choosing:

$$F(0) = \frac{1}{\delta} I = (GI) I;\ 0 < \delta \ll 1 \tag{2.55}$$

a typical value being $\delta = 0.001 (GI = 1000)$. It can be observed in the expression of $F(t+1)^{-1}$ given by (2.37) that the influence of this initial error decreases with the time. In this case one minimizes the following criterion:

$$\min_{\hat{\theta}(t)} J(t) = \sum_{i=1}^{t} [y(i) - \hat{\theta}^T(t)\phi(i-1)]^2 + [\theta - \hat{\theta}^T(0)]^T F(0)^{-1} [\theta - \hat{\theta}(0)]^T \tag{2.56}$$

A rigorous analysis shows nevertheless that for any positive definite matrix $F(0)[F(0) > 0]$,

$$\lim_{t\to\infty} \epsilon(t+1) = 0$$

The recursive least squares algorithm is an algorithm with a decreasing adaptation gain. This is clearly seen if the estimation of a single parameter is considered. In this case, $F(t)$ and $\phi(t)$ are scalars, and (2.53) becomes:

$$F(t+1) = \frac{F(t)}{1+\phi(t)^2 F(t)} \leq F(t); \qquad \phi(t), F(t) \in R^1$$

The same conclusion is obtained observing that $F(t+1)^{-1}$ is the output of an integrator which has as input $\phi(t)\phi^T(t)$. Since $\phi(t)\phi^T(t) \geq 0$, one conclude that if $\phi(t)\phi^T(t) > 0$ in the average, then $F(t)^{-1}$ will tends towards infinity, i.e., $F(t)$ will tends towards zero.
The recursive least squares algorithm in fact gives less and less weight to the new prediction errors and thus to the new measurements. Consequently, this type of variation of the adaptation gain is not suitable for the estimation of time-varying parameters, and other variation profiles for the adaptation gain must therefore be considered. Under certain conditions the adaptation gain is a measure of the evolution of the covariance of the parameter estimation error.
The least squares algorithm presented up to now for $\hat{\theta}(t)$ and $\phi(t)$ of dimension 2 may be generalized for any dimensions resulting from the description of discrete-time systems of the form:

$$y(t) = \frac{q^{-d}B(q^{-1})}{A(q^{-1})}u(t) \tag{2.57}$$

where:

$$A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_{n_A} q^{-n_A} \tag{2.58}$$
$$B(q^{-1}) = b_1 q^{-1} + \cdots + b_{n_B} q^{-n_B} \tag{2.59}$$

(2.57) can be written in the form:

$$y(t+1) = -\sum_{i=1}^{n_A} a_i y(t+1-i) + \sum_{i=1}^{n_B} b_i u(t-d-i+1) = \theta^T \phi(t) \tag{2.60}$$

in which:

$$\theta^T = [a_1, \cdots, a_{n_A}, b_1, \cdots, b_{n_B}] \tag{2.61}$$
$$\phi^T(t) = [-y(t)\cdots - y(t-n_A+1), u(t-d) \cdots u(t-d-n_B+1)] \tag{2.62}$$

The *a priori* adjustable predictor is given in the general case by:

$$\hat{y}^0(t+1) = -\sum_{i=1}^{n_A} \hat{a}_i(t) y(t+1-i) + \sum_{i=1}^{n_B} \hat{b}_1(t) u(t-d-i+1)$$
$$= \hat{\theta}^T(t)\phi(t) \tag{2.63}$$

in which:

$$\hat{\theta}^T(t) = [\hat{a}_1(t), \cdots, \hat{a}_{n_A}(t), \hat{b}_1(t), \cdots, \hat{b}_{n_B}(t)] \tag{2.64}$$

and for the estimation of $\hat{\theta}(t)$, the algorithm given in (2.51) through (2.54) is used, with the appropriate dimension for $\hat{\theta}(t)$, $\phi(t)$ and $F(t)$.

2.3 Choice of the Adaptation Gain

The recursive formula for the inverse of the adaptation gain $F(t+1)^{-1}$ given by (2.52) is generalized by introducing two weighting sequences $\lambda_1(t)$ and $\lambda_2(t)$, as indicated below:

$$F(t+1)^{-1} = \lambda_1(t)F(t)^{-1} + \lambda_2(t)\phi(t)\phi^T(t)$$
$$0 < \lambda_1(t) \leq 1 \; ; \; 0 \leq \lambda_2(t) < 2 \; ; \; F(0) > 0 \tag{2.65}$$

Note that $\lambda_1(t)$ and $\lambda_2(t)$ in (2.65) have the opposite effect. $\lambda_1(t) < 1$ tends to increase the adaptation gain (the gain inverse decreases); $\lambda_2(t) > 0$ tends to decrease the adaptation gain (the gain inverse increases). For the purpose of this manual only the influence of λ_1 will be discussed as it is the chosen parameter for implementing different adaptation schemes.
Using the *matrix inversion lemma* given by (2.44), one obtains from (2.65):

$$F(t+1) = \frac{1}{\lambda_1(t)} \left[F(t) - \frac{F(t)\phi(t)\phi^T(t)F(t)}{\frac{\lambda_1(t)}{\lambda_2(t)} + \phi^T(t)F(t)\phi(t)} \right] \tag{2.66}$$

Next, a certain number of choices for $\lambda_1(t)$ and their interpretations will be given.

A.1: Decreasing (vanishing) gain (RLS). In this case:

$$\lambda_1(t) = \lambda_1 = 1;\ \lambda_2(t) = 1 \tag{2.67}$$

and $F(t+1)^{-1}$ is given by (2.52), which leads to a decreasing adaptation gain. The minimized criterion is that of (2.31). This type of profile is suited to the estimation of the parameters of stationary systems.

A.2: Constant forgetting factor. In this case:

$$\lambda_1(t) = \lambda_1;\ 0 < \lambda_1 < 1;\ \lambda_2(t) = \lambda_2 = 1 \tag{2.68}$$

The typical values for λ_1 are:

$$\lambda_1 = 0.95 \text{ to } 0.99$$

The criterion to be minimized will be:

$$J(t) = \sum_{i=1}^{t} \lambda_1^{(t-i)} [y(i) - \hat{\theta}^T(t)\phi(i-1)]^2 \tag{2.69}$$

The effect of $\lambda_1(t) < 1$ is to introduce increasingly weaker weighting on the old data ($i < t$). This is why λ_1 is known as the *forgetting factor*. The maximum weight is given to the most recent error. This type of profile is suited to the estimation of the parameters of slowly time-varying systems. The use of a constant forgetting factor without the monitoring of the maximum value of $F(t)$ causes problems in adaptive regulation if the $\{\phi(t)\phi^T(t)\}$ sequence becomes null in the average (steady state case) because the adaptation gain will tend towards infinity. In this case:

$$F(t+i)^{-1} = (\lambda_1)^i F(t)^{-1}$$

and:

$$F(t+i) = (\lambda_1)^{-i} F(t)$$

For: $\lambda_1 < 1$, $\lim_{i\to\infty} (\lambda_1)^{-i} = \infty$ and $F(t+i)$ will become asymptotically unbounded.

A.3: Variable forgetting factor. In this case:

$$\lambda_2(t) = \lambda_2 = 1 \tag{2.70}$$

and the forgetting factor $\lambda_1(t)$ is given by:

$$\lambda_1(t) = \lambda_0\lambda_1(t-1) + 1 - \lambda_0\ ;\ 0 < \lambda_0 < 1 \tag{2.71}$$

The typical values being:

$$\lambda_1(0) = 0.95 \text{ to } 0.99\ ;\ \lambda_0 = 0.5 \text{ to } 0.99$$

($\lambda_1(t)$ can be interpreted as the output of a first order filter $(1-\lambda_0)/(1-\lambda_0 q^{-1})$ with a unitary steady state gain and an initial condition $\lambda_1(o)$).
Relation (2.71) leads to a forgetting factor that asymptotically tends towards 1. The criterion minimized will be:

$$J(t) = \sum_{i=1}^{t} \left[\prod_{j=1}^{t} \lambda_1(j-i) \right] [y(i) - \hat{\theta}^T(t)\phi(i-1)]^2 \tag{2.72}$$

As λ_1 tends towards 1 for large i, only the initial data are forgotten (the adaptation gain tends towards a decreasing gain).
This type of profile is highly recommended for the model identification of stationary systems, since it avoids a too rapid decrease of the adaptation gain, thus generally resulting in an acceleration of the convergence (by maintaining a high gain at the beginning when the estimates are at a great distance from the optimum).
Other types of evolution for $\lambda_1(t)$ can be considered. For example:

$$\lambda_1(t) = 1 - \frac{\phi^T(t)F(t)\phi(t)}{1 + \phi^T(t)F(t)\phi(t)}$$

This forgetting factor depends upon the input/output signals via $\phi(t)$. It automatically takes the value 1 if the norm of $\phi(t)\phi^T(t)$ becomes null. In the cases where the $\phi(t)$ sequence is such that the term $\phi^T(t)F(t)\phi(t)$ is significative with respect to one, the forgetting factor takes a lower value assuring good adaptation capabilities (this is related to the concept of "persistently exciting" signal).
Another possible choice is:

$$\lambda_1(t) = 1 - \alpha\frac{[\epsilon^0(t)]^2}{1 + \phi^T(t)F(t)\phi(t)} \; ; \; \alpha > 0$$

The forgetting factor tends towards 1, when the prediction error tends towards zero. Conversely, when a change occurs in the system parameters, the prediction error increases leading to a forgetting factor less than 1 in order to assure a good adaptation capability.

Choice of the initial gain $F(0)$.

The initial gain $F(0)$ is usually chosen as a diagonal matrix of the form given by (2.55) and, respectively,

$$F(0) = \begin{bmatrix} GI & & 0 \\ & \ddots & \\ 0 & & GI \end{bmatrix} \tag{2.73}$$

In the absence of initial information upon the parameters to be estimated (typical value of initial estimates $= 0$), a high initial gain (GI) is chosen. A typical value is $GI = 1000$ (but higher values can be chosen).
If an initial parameter estimation is available (resulting for example from a previous identification), a low initial gain is chosen. In general, in this case $GI \leq 1$.
Since in standard RLS the adaptation gain decreases as the true model parameters are approached (a significant measurement is its trace), the adaptation gain may be interpreted as a measurement of the accuracy of the parameter estimation (or prediction). This explains the choices of $F(0)$ proposed above. Note that under certain hypotheses, $F(t)$ is effectively a measurement of the quality of the estimation.
This property can give indications upon the evolution of a parameter estimation procedure. If the trace of $F(t)$ did not decrease significantly, the parameter estimation is in general poor. This may occur in system identification when the level and type of input used are not appropriate.

2.4 Some Remarks on the Parameter Adaptation Algorithms

The parameter adaptation algorithms (PAA) presented up to now (integral type) have the form ($\lambda_1 = 1; \lambda_2 = 1$):

$$\hat{\theta}(t+1) = \hat{\theta}(t) + F(t+1)\phi(t)\epsilon^0(t+1) = \hat{\theta}(t) + \frac{F(t)\phi(t)\epsilon^0(t+1)}{1 + \phi^T(t)F(t)\phi(t)}$$

where $\hat{\theta}(t)$ is the vector of estimated parameters and $F(t+1)\phi(t)\epsilon^0(t+1)$ represents the correcting term at each sample.
$F(t)$ is the adaptation gain (constant or time-varying), $\phi(t)$ is the observation vector and $\epsilon^0(t+1)$ is the *a priori* prediction error (or in general the adaptation error), i.e., the difference between the measured output at the instant $t+1$ and the predicted output at $t+1$ based on the knowledge of $\hat{\theta}(t)$.

There is always a relationship between the *a priori* prediction (adaptation) error $\epsilon^0(t+1)$ and the *a posteriori* prediction (adaptation) error $\epsilon(t+1)$ defined on the basis of the knowledge of $\hat{\theta}(t+1)$. This relation is:

$$\epsilon(t+1) = \frac{\epsilon^0(t+1)}{1+\phi^T(t)F(t)\phi(t)}$$

Several types of updating formulas for the adaptation gain can be used in connection with the type of the parameter estimation problem to be solved (systems with fixed or time-varying parameters), with or without availability of initial information for the parameter to be estimated.
The PAA examined up to now are based on:

- minimization of a one step ahead criterion
- off-line minimization of a criterion over a finite horizon followed by a sequential minimization
- sequential minimization of a criterion over a finite horizon starting at $t = 0$
- relationship with the Kalman predictor.

However, from the point of view of real-time identification and adaptive control, the parameter adaptation algorithms are supposed to operate on a very large number of measurements ($t \Longrightarrow \infty$). Therefore, it is necessary to examine the properties of parameter adaptation algorithms as $t \Longrightarrow \infty$. Specifically, one should study the conditions which guarantee:

$$\lim_{t\to\infty} \epsilon(t+1) = 0$$

This corresponds to the study of the stability of parameter adaptation algorithms. Conversely, other parameter adaptation algorithms will be derived from the stability condition given above.

2.5 Practical aspects of recursive open loop identification

In this section, the algorithms available in iREG for open loop plant model recusrive identification will be presented. From a practical point of view, plant model identification is a key starting point for designing a high perfomance linear controller.
Identification of dynamic systems is an experimental approach for determining a dynamic model of a system. It includes four steps:

1. Input-output data acquisition under an experimental protocol.
2. Selection (or estimation) of the model complexity (structure).
3. Estimation of the model parameters.
4. Validation of the identified model (structure of the model and values of the parameters).

As it will be seen, the algorithms which will be used for parameter estimation will depend on the assumptions made on the noise disturbing the measurements, assumptions which have to be confirmed by the model validation ([6], [4], [5]). It is important to emphasize that no one single *plant + disturbance* structure exists that can describe all the situations encountered in practice. Furthermore, there is no parameter estimation algorithms that may be used with all possible plant + disturbance structures such that the estimated parameters are always unbiased. Furthermore, due to the lack of *a priori* information, the input-output data acquisition protocol may be initially inappropriate.
All these causes may lead to identified models which do not pass the validation test, and therefore the identification should be viewed as an iterative process as illustrated in Figure 2.5.

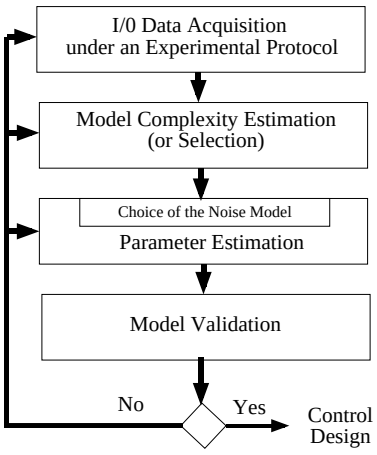


Figure 2.5: The iterative process of identification

Tables 2.1-2.3 summarizes a number of significant recursive parameter estimation techniques. They all use PAA of the form:

$$\hat{\theta}(t+1) = \hat{\theta}(t) + F(t)\Phi(t)\nu(t+1) \tag{2.74}$$

$$F(t+1)^{-1} = \lambda_1(t)F(t) + \lambda_2(t)\Phi(t)\Phi^T(t) \tag{2.75}$$

$$0 < \lambda_1(t) \leq 1 \quad ; \quad 0 \leq \lambda_2(t) < 2; F(0) > 0$$

$$F^{-1}(t) > \alpha F^{-1}(0) \quad ; \quad 0 < \alpha < \infty$$

$$\nu(t+1) = \frac{\nu^0(t+1)}{1 + \Phi^T(t)F(t)\Phi(t)} \tag{2.76}$$

2.5.1 Validation of the Models Identified with Type I Methods

This section is concerned with the validation of models identified using identification methods based on the *whitening* of the prediction error.
If the residual prediction error is a white noise sequence, in addition to obtaining unbiased parameter estimates, this also means that the identified model gives the best prediction for the plant output in the sense that it minimizes the variance of the prediction error. On the other hand, since the residual error is white and a white noise is not correlated with any other variable, then all the correlations between the input and the output of the plant are represented by the identified model and what remains unmodelled does not depend on the input. The principle of the validation method is as follows:

- If the plant + disturbance structure chosen is correct, i.e., representative of reality.
- If an appropriate identification method for the structure chosen has been used.
- If the degrees of the polynomials $A(q^{-1}), B(q^{-1}), C(q^{-1})$ and the value of d (delay) have been correctly chosen (the plant model is in the model set).

Then the prediction error $\epsilon(t)$ asymptotically tends toward a white noise, which implies:

$$\lim_{t\to\infty} E\{\epsilon(t)\epsilon(t-i)\} = 0; \qquad i = 1, 2, 3\cdots \; ; \; -1, -2, -3\cdots$$

The validation method implements this principle. It is made up of several steps:

1 - Creation of an I/O file for the identified model (using the same input sequence as for the system).

2 - Creation of a prediction error file for the identified model (minimum 100 data).

3 - *Whiteness* (uncorrelatedness) test on the prediction errors sequence (also known as residual prediction errors).

Table 2.1: Recursive identification algorithms

		Recursive Least Squares (RLS)	Extended Least Squares (RELS)	Output Error with Extended Prediction Model (XOLOE)
Plant + Noise Model		$y = \frac{q^{-d}B}{A}u + \frac{1}{A}e$	$y = \frac{q^{-d}B}{A}u + \frac{C}{A}e$	$y = \frac{q^{-d}B}{A}u + \frac{C}{A}e$
Adjustable Parameter Vector		$\hat{\theta}^T(t) = [\hat{a}^T(t), \hat{b}^T(t)]$ $\hat{a}^T(t) = [\hat{a}_1 \dots \hat{a}_{n_A}]$ $\hat{b}^T(t) = [\hat{b}_1 \dots \hat{b}_{n_B}]$	$\hat{\theta}^T(t) = [\hat{a}^T(t), \hat{b}^T(t), \hat{c}^T(t)]$ $\hat{a}^T(t) = [\hat{a}_1 \dots \hat{a}_{n_A}]$ $\hat{b}^T(t) = [\hat{b}_1 \dots \hat{b}_{n_B}]$ $\hat{c}^T(t) = [\hat{c}_1 \dots \hat{c}_{n_C}]$	$\hat{\Theta}^T(t) = [\hat{a}^T(t), \hat{b}^T(t), \hat{h}^T(t)]$ $\hat{a}^T(t) = [\hat{a}_1 \dots \hat{a}_{n_A}]$ $\hat{b}^T(t) = [\hat{b}_1 \dots \hat{b}_{n_B}]$ $\hat{h}^T(t) = [\hat{h}_1 \dots \hat{h}_{n_H}]$ $n_H = \max(n_A, n_C)$
Predictor Regressor Vector		$\phi^T(t) = [-y(t) \dots - y(t - n_A + 1),$ $u(t-d) \dots u(t - d - n_B + 1)]$	$\phi^T(t) = [-y(t) \dots - y(t - n_A + 1),$ $u(t-d) \dots u(t - d - n_B + 1),$ $\epsilon(t) \dots \epsilon(t - n_C + 1)]$	$\phi^T(t) = [-\hat{y}(t) \dots - \hat{y}(t - n_A + 1),$ $u(t-d) \dots u(t - d - n_B + 1),$ $\epsilon(t) \dots \epsilon(t - n_C + 1)]$
Predictor Output	a priori	$\hat{y}^0(t+1) = \hat{\theta}^T(t)\phi(t)$		
	a posteriori	$\hat{y}(t+1) = \hat{\theta}^T(t+1)\phi(t)$		
Prediction Error	a priori	$\epsilon^0(t+1) = y(t+1) - \hat{y}^0(t+1)$		
	a posteriori	$\epsilon(t+1) = y(t+1) - \hat{y}(t+1)$		
Adaptation Error		$\nu^\circ(t+1) = \epsilon^\circ(t+1)$		
Observation Vector		$\Phi(t) = \phi(t)$		

Table 2.2: Recursive identification algorithms

		Recursive Maximum Likelihood (RML)	Generalized Least Squares (GLS)	Output Error (OE)
Plant + Noise Model		$y = \frac{q^{-d}B}{A}u + \frac{C}{A}e$	$y = \frac{q^{-d}B}{A}u + \frac{C}{AD}e$	$y = \frac{q^{-d}B}{A}u + v$
Adjustable Parameter Vector		$\hat{\theta}^T(t) = [\hat{a}^T(t), \hat{b}^T(t), \hat{c}^T(t)]$ $\hat{a}^T(t) = [\hat{a}_1 \dots \hat{a}_{n_A}]$ $\hat{b}^T(t) = [\hat{b}_1 \dots \hat{b}_{n_B}]$ $\hat{c}^T(t) = [\hat{c}_1 \dots \hat{c}_{n_C}]$	$\hat{\theta}^T(t) = [\hat{a}^T(t), \hat{b}^T(t), \hat{c}^T(t), \hat{d}^T(t)]$ $\hat{a}^T(t) = [\hat{a}_1 \dots \hat{a}_{n_A}]$ $\hat{b}^T(t) = [\hat{b}_1 \dots \hat{b}_{n_B}]$ $\hat{c}^T(t) = [\hat{c}_1 \dots \hat{c}_{n_C}]$ $\hat{d}^T(t) = [\hat{d}_1 \dots \hat{d}_{n_D}]$	$\hat{\theta}^T(t) = [\hat{a}^T(t), \hat{b}^T(t)]$ $\hat{a}^T(t) = [\hat{a}_1 \dots \hat{a}_{n_A}]$ $\hat{b}^T(t) = [\hat{b}_1 \dots \hat{b}_{n_B}]$
Predictor Regressor Vector		$\phi^T(t) = [-y(t) \dots - y(t-n_A+1),$ $u(t-d) \dots u(t-d-n_B+1),$ $\epsilon(t) \dots \epsilon(t-n_C+1)]$	$\phi^T(t) = [-y(t) \dots - y(t-n_A+1),$ $u(t-d) \dots u(t-d-n_B+1),$ $\epsilon(t) \dots \epsilon(t-n_C+1),$ $-\alpha(t) \dots - \alpha(t-n_D+1)]$ $\alpha(t) = \hat{A}(t)y(t) - \hat{B}^*(t)u(t-d-1)$	$\phi^T(t) = [-\hat{y}(t) \dots - \hat{y}(t-n_A+1),$ $u(t-d) \dots u(t-d-n_B+1)]$
Predictor Output	a priori	$\hat{y}^0(t+1) = \hat{\theta}^T(t)\phi(t)$		
	a posteriori	$\hat{y}(t+1) = \hat{\theta}^T(t+1)\phi(t)$		
Prediction Error	a priori	$\epsilon^0(t+1) = y(t+1) - \hat{y}^0(t+1)$		
	a posteriori	$\epsilon(t+1) = y(t+1) - \hat{y}(t+1)$		
Adaptation Error		$\nu^\circ(t+1) = \epsilon^\circ(t+1)$		
Observation Vector		$\Phi(t) = \frac{1}{\hat{C}(t,q^{-1})}\phi(t)$ $\hat{C}(t,q^{-1}) = 1 + \hat{c}_1(t)q^{-1} + \dots$	$\Phi(t) = \phi(t)$	$\Phi(t) = \phi(t)$

Table 2.3: Recursive identification algorithms

		Output Error with (Adaptive) Filtered Observations (A)FOLOE	Instrumental Variable with Auxiliary Model (VI_MAUX)	Box Jenkins (BJ)
Plant + Noise Model		$y = \frac{q^{-d}B}{A}u + v$	$y = \frac{q^{-d}B}{A}u + v$	$y = \frac{q^{-d}B}{A}u + \frac{C}{D}e$
Adjustable Parameter Vector		$\hat{\theta}^T(t) = [\hat{a}^T(t), \hat{b}^T(t)]$ $\hat{a}^T(t) = [\hat{a}_1 \dots \hat{a}_{n_A}]$ $\hat{b}^T(t) = [\hat{b}_1 \dots \hat{b}_{n_B}]$	$\hat{\theta}^T(t) = [\hat{a}^T(t), \hat{b}^T(t)]$ $\hat{a}^T(t) = [\hat{a}_1 \dots \hat{a}_{n_A}]$ $\hat{b}^T(t) = [\hat{b}_1 \dots \hat{b}_{n_B}]$	$\hat{\theta}^T(t) = [\hat{a}^T(t), \hat{b}^T(t), \hat{c}^T(t), \hat{d}^T(t)]$ $\hat{a}^T(t) = [\hat{a}_1 \dots \hat{a}_{n_A}]$ $\hat{b}^T(t) = [\hat{b}_1 \dots \hat{b}_{n_B}]$ $\hat{c}^T(t) = [\hat{c}_1 \dots \hat{c}_{nC}]$ $\hat{d}^T(t) = [\hat{d}_1 \dots \hat{d}_{n_D}]$
Predictor Regressor Vector		$\phi^T(t) = [-\hat{y}(t) \dots - \hat{y}(t - n_A + 1),$ $u(t-d) \dots u(t-d-n_B+1)]$	$\phi^T(t) = [-y(t) \dots - y(t - n_A + 1),$ $u(t-d) \dots u(t-d-n_B+1)]$	$\phi^T(t) = [-\hat{x}(t) \dots - \hat{x}(t - n_A + 1),$ $u(t-d) \dots u(t-d-n_B+1)$ $\hat{v}(t) \dots \hat{v}(t - n_B + 1)$ $-\hat{w}(t) \dots - \hat{w}(t - n_B + 1)]$ $\hat{x}(t) = \theta[1 : n_A + n_B]^T(t)\phi[1 : n_A + n_B](t)$ $\hat{v}(t) = y(t) - \hat{y}(t)$ $\hat{w}(t) = y(t) - \hat{x}(t)$
Predictor Output	a priori	$\hat{y}^0(t+1) = \hat{\theta}^T(t)\phi(t)$	$\hat{y}^0(t+1) = \hat{\theta}^T(t)\phi(t)$	$\hat{y}^0(t+1) = \hat{\theta}^T(t)\phi(t)$
	a posteriori	$\hat{y}(t+1) = \hat{\theta}^T(t+1)\phi(t)$	$\hat{y}(t+1) = \hat{\theta}^T(t+1)\phi(t)$	$\hat{y}(t+1) = \hat{\theta}^T(t+1)\phi(t)$
Prediction Error	a priori	$\epsilon^0(t+1) = y(t+1) - \hat{y}^0(t+1)$		
	a posteriori	$\epsilon(t+1) = y(t+1) - \hat{y}(t+1)$		
Adaptation Error		$\nu^\circ(t+1) = \epsilon^\circ(t+1)$	$\nu^\circ(t+1) = \epsilon^\circ(t+1)$	$\nu^\circ(t+1) = \epsilon^\circ(t+1)$
Observation Vector		$\Phi(t) = \frac{1}{L(q^{-1})}\phi(t)$ $\left(\Phi(t) = \frac{1}{\hat{A}(t,q^{-1})}\phi(t)\right)$ $\left(\hat{A}(t,q^{-1}) = 1 + \hat{a}_1(t)q^{-1} + \dots\right)$	$\Phi^T(t) = [-y_{IV}(t) \dots - y_{IV}(t - n_A + 1)$ $u(t-d) \dots u(t-n_B-d+1)]$ $y_{IV}(t) = \hat{\theta}^T(t)\Phi(t-1)$	$\Phi(t) = \phi(t)$

Table 2.4: Confidence intervals for whiteness tests

Level of significance	Validation criterion	$N = 128$	$N = 256$	$N = 512$
3%	$\frac{2.17}{\sqrt{N}}$	0.192	0.136	0.096
5%	$\frac{1.96}{\sqrt{N}}$	0.173	0.122	0.087
7%	$\frac{1.808}{\sqrt{N}}$	0.16	0.113	0.08

Whiteness test

Let $\{\epsilon(t)\}$ be the centered sequence of the residual prediction errors (centered: measured value - mean value). One computes:

$$R(0) = \frac{1}{N}\sum_{t=1}^{N}\epsilon^2(t)\ ;\ RN(0) = \frac{R(0)}{R(0)} = 1 \tag{2.77}$$

$$R(i) = \frac{1}{N}\sum_{t=1}^{N}\epsilon(t)\epsilon(t-i)\ ;\ RN(i) = \frac{R(i)}{R(0)} \tag{2.78}$$

$$i = 1, 2, 3 \cdots n_A \cdots$$

with $i_{max} \geq n_A$ [degree of polynomial $A(q^{-1})$], which are estimations of the (normalized) autocorrelations.
If the residual prediction error sequence is perfectly white (theoretical situation), and the number of samples is very large ($N \to \infty$), then $RN(0) = 1$; $RN(i) = 0,\ i \geq 1$[3].
In real situations, however, this is never the case (i.e., $RN \neq 0; i \geq 1$), therefore, one considers as a practical validation criterion (extensively tested on applications):

$$RN(0) = 1\ ;\ \mid RN(i) \mid \leq \frac{2.17}{\sqrt{N}}\ ;\ i \geq 1 \tag{2.79}$$

where N is the number of samples.
This confidence interval corresponds to a 3% level of significance of the hypothesis test for Gaussian distribution. Sharper confidence intervals can be defined. Table 2.4 gives the values of the validation criterion for various N and various levels of significance.
The following remarks are important:

- An acceptable identified model has in general:

$$\mid RN(i) \mid \leq \frac{1.8}{\sqrt{N}} \cdots \frac{2.17}{\sqrt{N}}\ ;\ i \geq 1$$

- If several identified models have the same complexity (number of parameters), one chooses the model given by the methods that lead to the smallest $\mid RN(i) \mid$.
- A *too good* validation criterion indicates that model simplifications may be possible.
- For simplicity's sake, one can consider as a basic practical numerical value for the validation criterion value:

$$\mid RN(i) \mid \leq 0.15\ ;\ i \geq 1$$

- If the level of the residual prediction errors is very low compared to the output level (let us say more than 60 dB), the whiteness tests lose their signification.

[3] Conversely, for Gaussian data, uncorrelation implies independence. In this case, $RN(i) = 0, i \geq 1$ implies independence between $\epsilon(t), \epsilon(t-1) \cdots$, i.e., the sequence of residuals $\{\epsilon(t)\}$ is a Gaussian white noise.

2.5.2 Validation of the Models Identified with Type II Methods

This section is concerned with the validation of models obtained using the identification methods based on the decorrelation between the observations and the prediction errors. The uncorrelation between the observations and the prediction error leads to an unbiased parameter estimation. However, since the observations include the predicted output which depends on the input, the uncorrelation between the residual prediction error and the observations implies also that the residual prediction error is uncorrelated with the input. The interpretation of this fact is that the residual prediction error does not contain any information depending upon the input, i.e., all the correlations between the input and the output of the plant are represented by the identified model. The principle of the validation method is as follows:

- If the disturbance is independent of the input ($\Longrightarrow E\{w(t)u(t)\} = 0$).
- If the *model + disturbance* structure chosen is correct, i.e., representative of the reality.
- If an appropriate identification method has been used for the chosen structure.
- If the degrees of the polynomials $A(q^{-1}), B(q^{-1})$ and the value of d (delay) have been correctly chosen (the plant model is in the model set).

then the predicted outputs $\hat{y}(t-1), \hat{y}(t-2)\cdots$ generated by a model of the form (output error type predictor):

$$\hat{A}(q^{-1})\hat{y}(t) = q^{-d}\hat{B}(q^{-1})u(t) \tag{2.80}$$

and the prediction error $\epsilon(t)$ are asymptotically uncorrelated, which implies:

$$E\left\{\epsilon(t)\hat{y}(t-i)\right\} \approx \frac{1}{N}\sum_{t=1}^{N}\epsilon(t)\hat{y}(t-i) = 0$$
$$i = 1, 2, 3$$

The validation method implements this principle. It is made up of several steps:

1. Creation of an I/O file for the identified model (using the same input sequence as for the system).
2. Creation of files for the sequences $\{y(t)\}; \{\hat{y}(t)\}; \{\epsilon(t)\}$ (system output, model output, residual output prediction error). These files must contain at least 100 data in order that the test be significant.
3. *Uncorrelation* test between the residual output prediction error sequence and the delayed prediction model output sequences.

Uncorrelation test

Let $\{y(t)\}$ and $\{\hat{y}(t)\}$ be the centered output sequences of the plant and of the identified model, respectively. Let $\{\epsilon(t)\}$ be the centered sequence of the residual output (prediction) errors (centered = measured values - mean values). One computes:

$$R(i) = \frac{1}{N}\sum_{t=1}^{N}\epsilon(t)\hat{y}(t-i)\ ;\ i = 0, 1, 2, \cdots, n_A \cdots \tag{2.81}$$

$$RN(i) = \frac{R(i)}{\left[\left(\frac{1}{N}\sum_{t=1}^{N}\hat{y}^2(t)\right)\left(\frac{1}{N}\sum_{t=1}^{N}\epsilon^2(t)\right)\right]^{1/2}}$$

$$i = 0, 1, 2, \cdots, n_A \cdots \tag{2.82}$$

which are estimations of the (normalized) cross-correlations (note that $RN(0) \neq 1$), and n_A is the degree of polynomial $A(q^{-1})$.
If $\epsilon(t)$ and $\hat{y}(t-i), i \geq 1$ are perfectly uncorrelated (theoretical situation), then:

$$RN(i) = 0\ ;\ i = 1, 2, \cdots, n_A \cdots$$

In practice, one considers as a practical validation criterion:

$$\mid RN(i) \mid \leq \frac{2.17}{\sqrt{N}} \; ; \; i \geq 1$$

where N is the number of samples. Table 2.4 can be used to find the numerical value of the validation criterion for various N and the level of signification of the test.
All the comments made in the previous section apply also in this case. In particular, the basic practical numerical value for the validation criterion, which is:

$$\mid RN(i) \mid < 0.15 \; ; \; i \geq 1$$

is worth remembering.
This test is also used when one would like to compare models identified with Type I method, with models identified with Type II method.

2.5.3 Selection of the Pseudo Random Binary Sequence

The correct parameter estimation requires the use of a rich signal (persistently exciting signal). Pseudo-random binary sequences (PRBS) offer on the one hand a signal with a large frequency spectrum approaching the white noise and, on the other hand, they have a constant magnitude. This allows to define precisely the level of the instantaneous stress on the process (or actuator).
PRBS are sequences of rectangular pulses, modulated in width, that approximate a discrete-time white noise and thus have a spectral content *rich* in frequencies. They owe their name *pseudo random* to the fact that they are characterized by a *sequence length* within which the variations in pulse width vary randomly, but that over a large time horizon, they are periodic, the period being defined by the length of the sequence. In the practice of system identification, one generally uses just one complete sequence.
The PRBS are generated by means of shift registers with feedback (implemented in hardware or software). The maximum length of a sequence is $2^N - 1$, in which N is the number of cells of the shift register. Figure 2.6

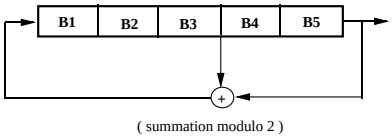


Figure 2.6: Generation of a PRBS of length $2^5 - 1 = 31$ sampling periods

presents the generation of a PRBS of length $31 = 2^5 - 1$ obtained by means of a 5-cells shift register.
Note that at least one of the N cells of the shift register should have an initial logic value different from zero (one generally takes all the initial values of the N cells equal to the logic value 1).
Table 2.5 gives the structure enabling maximum length PRBS to be generated for different numbers of cells.
Note also a very important characteristic element of the PRBS: *the maximum duration of a PRBS impulse is equal to N (number of cells).*
This property is to be considered when choosing a PRBS for system identification.
In order to correctly identify the steady-state gain of the plant dynamic model, the duration of at least one of the pulses (e.g., the maximum duration pulse) must be greater than the rise time t_R of the plant. The maximum duration of a pulse being $N.T_s$, the following condition results:

$$N.T_s > t_R \tag{2.83}$$

which is illustrated in Figure 2.8. From condition (2.83), one determines N and therefore the length of the sequence, which is $2^N - 1$.
Furthermore, in order to cover the entire frequency spectrum generated by a particular PRBS, the length of a test must be at least equal to the length of the sequence. In a large number of cases, the duration of the test L is chosen equal to the length of the sequence. If the duration of the test is specified, it must therefore be ensured that:

$$(2^N - 1).T_s < L \quad (L = \text{test duration}) \tag{2.84}$$

Table 2.5: Generation of maximum length PRBS

Number of Cells N	Sequence Length $L = 2^N - 1$	Bits Added B_i and B_j
2	3	1 and 2
3	7	1 and 3
4	15	3 and 4
5	31	3 and 5
6	63	5 and 6
7	127	4 and 7
8	255	2,3,4 and 8
9	511	5 and 9
10	1023	7 and 10

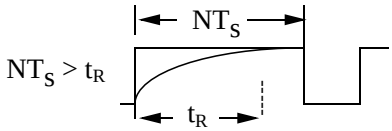


Figure 2.7: Choice of a maximum duration of a pulse in a PRBS

Note that the condition (2.83) can result in fairly large values of N corresponding to sequence lengths of prohibitive duration, either because T_s is very large, or because the system to be identified may well evolve during the duration of the test. This is why, in a large number of practical situations, a submultiple of the sampling frequency is chosen as the clock frequency for the PRBS. If:

$$f_{PRBS} = \frac{f_s}{p} \; ; \; p = 1, 2, 3 \cdots \tag{2.85}$$

then condition 2.83 becomes:

$$p.N.T_s > t_R \tag{2.86}$$

This approach is more interesting than the increase of the sequence length by increasing N in order to satisfy (2.83).
Note that dividing the clock frequency of the PRBS will reduce the frequency range corresponding to a constant spectral density in the high frequencies while augmenting the spectral density in the low frequencies. In general, this will not affect the quality of identification, either because in many cases when this solution is considered, the plant to be identified has a low band pass or because the effect or the reduction of the signal/noise ratio at high frequencies can be compensated by the use of appropriate identification techniques. However, it is recommended to choose $p \leq 4$.
Figure 2.8 shows the spectral density of PRBS sequences generated with $N = 8$ for $p = 1, 2, 3$. As one can see, the energy of the spectrum is reduced in the high frequencies and augmented in the lower frequencies. Furthermore, for $p = 3$ a whole occurs at $f_s/3$.
Up to now, we have been concerned only with the choice of the length and clock frequency of the PRBS; however, the magnitude of the PRBS must also be considered. Although the magnitude of the PRBS may be very low, it should lead to output variations larger than the residual noise level. If the signal/noise ratio is too low, the length of the test must be augmented in order to obtain a satisfactory parameter estimation.
Note that in a large number of applications, the significant increase in the PRBS level may be undesirable in view of the nonlinear character of the plants to be identified (we are concerned with the identification of a linear model around an operating point).

2.5.4 Model Order Selection

In the absence of a clear *a priori* information upon the plant model structure, two approaches can be used to select the appropriate orders for the values of d, n_A, n_B characterizing the input-output plant model.

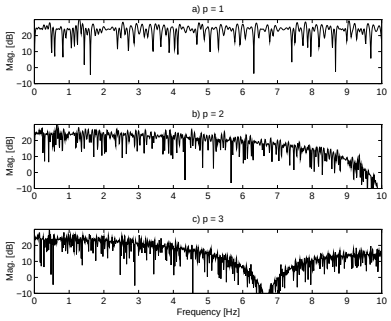


Figure 2.8: Spectral density of a PRBS sequence ($f_s = 20$ Hz), a) N=8, p=1, b) N=8, p=2, c) N=8, p=3

a) a practical approach based on trial and error

b) estimation of d, n_A, n_B directly from data.

Even in the case of using an estimation of the orders directly from data it is useful to see to what extend the two approaches are coherent.
We will discuss the two approaches next.

2.5.5 A Practical Approach for Model Order Selection

The *plant + disturbance* model to be identified is of the form:

$$A(q^{-1})y(t) = q^{-d}B(q^{-1})u(t) + w(t)$$

where, according to the structures chosen, one has:

$$\begin{aligned} w(t) &= e(t) \\ w(t) &= A(q^{-1})v(t); (v(t) \text{ and } u(t) \text{ are independent}) \\ w(t) &= C(q^{-1})e(t) \\ w(t) &= \frac{C(q^{-1})}{D(q^{-1})}e(t) \end{aligned}$$

The degrees of the polynomials, $A(q^{-1}), B(q^{-1}), C(q^{-1})$ and $D(q^{-1})$ are respectively n_A, n_B, n_C and n_D. In order to start the parameter estimation methods n_A, n_B and d must be specified. For the methods which estimate also the noise model, one needs in addition to specify n_C and n_D.

***A priori* choice of n_A** Two cases can be distinguished:
1. Industrial plant (temperature, control, flow rate, concentration, and so on). For this type of plant in general:

$$n_A \leq 3$$

and the value $n_A = 2$, which is very typical, is a good starting value to choose.
2. Electromechanical systems. n_A results from the structural analysis of the system.

Example: Flexible Robot Arm with two vibration modes
In this case, $n_A = 4$ is chosen, since a second-order is required, to model a vibratory mode.

Initial choice of d and n_B If no knowledge of the time delay is available, $d = 0$ is chosen as an initial value. If a minimum value is known, an initial value $d = d_{\min}$ is chosen.
If the time delay has been underestimated, during identification the first coefficients of $B(q^{-1})$ will be very low. n_B must then be chosen so that it can both indicate the presence of the time delays and identify the transfer function numerator. $n_B = (d_{\max} - d_{\min}) + 2$ is then chosen as the initial value.
At least two coefficients are required in $B(q^{-1})$ because of the *fractional* delay which is often present in applications. If the time delay is known, $n_B \geq 2$ is chosen, but 2 remains a good initial value.

Determination of time delay d (first approximation) One identifies using the RLS (Recursive Least Squares). The estimated numerator will be of the form:

$$\hat{B}(q^{-1}) = \hat{b}_1 q^{-1} + \hat{b}_2 q^{-2} + \hat{b}_3 q^{-3} + \cdots$$

If:

$$\mid \hat{b}_1 \mid < 0.15 \mid \hat{b}_2 \mid$$

$b_1 \approx 0$ is considered and time delay d is increased by $1 : d = d_{\min} + 1$ [since if $b_1 = 0, B(q^{-1}) = q^{-1}(b_2 q^{-1} + b_3 q^{-2})$]. If:

$$\mid \hat{b}_i \mid < 0.15 \mid \hat{b}_{di+1} \mid \; ; \; i = 1, 2, \cdots, d_i$$

time delay d is increased by $d_i : d = d_{in} + d_i$. After these modifications, identification is restarted.

Determination of the $(n_A)_{\max}$ and $(n_B)_{\max}$ The aim is to obtain the simplest possible identified model that verifies the validation criteria. This is linked on the one hand to the complexity of the controller (which will depend on n_A and n_B) but equally to the robustness of the identified model with respect to the operating conditions.
A first approach to estimate the values of $(n_A)_{\max}$ and $(n_B)_{\max}$ is to use the RLS and to study the evolution of the variance of the residual prediction errors, i.e., the evolution of:

$$R(0) = E\left\{\epsilon^2(t)\right\} = \frac{1}{N}\sum_{t=1}^{N} \epsilon(t)^2$$

as a function of the value of $n_A + n_B$. A typical curve is given in Figure 5.8.1.

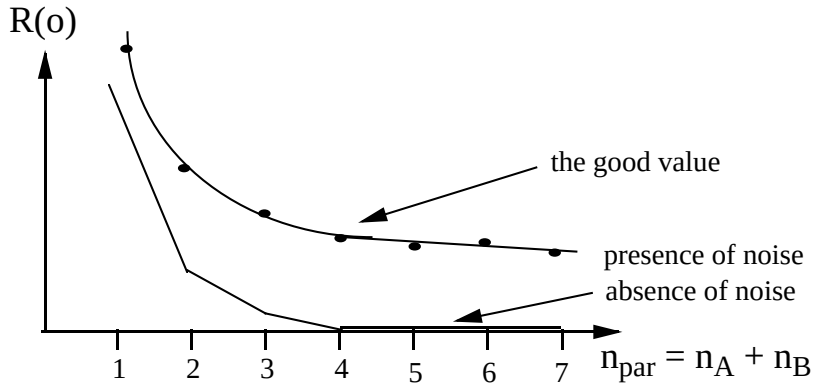


Figure 2.9: Evolution of the variance of residual errors as a function of the number of model parameters

In theory, if the example considered is simulated and noise-free, the curve should present a neat elbow followed by a horizontal segment, which indicates that the increase in parameter number does not improve the performance. In practice, this elbow is not neat because measurement noise is present.
The practical test used for determining $n_A + n_B$ is the following: consider first n_A, n_B and the corresponding variance of the residual errors $R(0)$.
Consider now $n'_A = n_A + 1$, n_B and the corresponding variance of the residual errors $R'(0)$. If:

$$R'(0) \geq 0.8R(0)$$

it is unwise to increase the degree of n_A (same test with $n'_B = n_B + 1$).
With the choice that results for n_A and n_B, the model identified by the RLS does not necessarily verify the validation criterion. Therefore, while keeping the values of n_A and n_B, other structures and methods must be tried out in order to obtain a *valid* model. If after all the methods have been tried, none is able to give a model that satisfies the validation criterion, then n_A and n_B must be increased.
For a more detailed discussion of various procedures for the estimation of $(n_A)\max$ and $(n_B)\max$.

Initial Choice of n_C and n_D (Noise Model)

As a rule, $n_C = n_D = n_A$ is chosen.

2.5.6 Direct Order Estimation from Data

To introduce the problem of order estimation from data, we will start with an example: Assume that the plant model can be described by:

$$y(t) = -a_1 y(t-1) + b_1 u(t-1) \tag{2.87}$$

and that the data are noise free. The order of this model is $n = n_A = n_B = 1$. Question: Is any way to test from data if the order assumption is correct? To do so, construct the following matrix:

$$\begin{bmatrix} y(t) & \vdots & y(t-1) & u(t-1) \\ y(t-1) & \vdots & y(t-2) & u(t-2) \\ y(t-2) & \vdots & y(t-3) & u(t-3) \end{bmatrix} = \begin{bmatrix} Y(t) & \vdots & R(1) \end{bmatrix} \tag{2.88}$$

Clearly, if the order of the model given in Eq. (2.87) is correct, the vector $Y(t)$ will be a linear combination of the columns of $R(1)$ ($Y(t) = R(1)\theta$ with $\theta^T = [-a_1, b_1]$) and the rank of the matrix will be 2 (instead of 3). If the plant model is of order 2 or higher, the matrix (2.88) will be full rank. Of course, this procedure can be extended for testing the order of a model by testing the rank of the matrix $[Y(t), R(\hat{n})]$ where:

$$\begin{aligned} R(\hat{n}) &= [Y(t-1), U(t-1), Y(t-2), U(t-2) \\ &\quad \cdots Y(t-\hat{n}), U(t-\hat{n})] && (2.89) \\ Y^T(t) &= [y(t), y(t-1) \cdots] && (2.90) \\ U^T(t) &= [u(t), u(t-1) \cdots] && (2.91) \end{aligned}$$

Unfortunately, as a consequence of the presence of noise, this procedure cannot directly be applied in practice. A more practical approach results from the observation that the rank test problem can be approached by the searching of $\hat{\theta}$ which minimize the following criterion for an estimated value of the order $\hat{n}$.

$$V_{LS}(\hat{n}, N) = \min_{\hat{\theta}} \frac{1}{N} \|Y(t) - R(\hat{n})\hat{\theta}\|^2 \tag{2.92}$$

where N is the number of data. But this criterion is nothing else than an equivalent formulation of the least squares. If the conditions for unbiased estimation using least squares are fulfilled, (2.92) is an efficient way for assessing the order of the model since $V_{LS}(\hat{n}) - V_{LS}(\hat{n}+1) \to 0$ when $\hat{n} \geq n$.
In the mean time, the objective of the identification is to estimate lower order models (parsimony principle) and therefore, it is reasonable to add in the criterion (2.92) a term which penalizes the complexity of the model. Therefore, the criterion for order estimation will take the form:

$$CV_{LS}(\hat{n}, N) = V_{LS}(\hat{n}, N) + S(\hat{n}, N) \tag{2.93}$$

where typically:

$$S(\hat{n}, N) = 2\hat{n}X(N) \tag{2.94}$$

$X(N)$ in (2.94) is a function that decreases with N. For example, in the so called $BIC_{LS}(\hat{n}, N)$ criterion, $X(N) = \frac{\log N}{N}$ (other choices are possible) and the order $\hat{n}$ is selected as the one which minimizes CV_{LS} given by (2.93). Unfortunately, the results are unsatisfactory in practice because in the majority of situations, the conditions for unbiased parameter estimation using least squares are not fulfilled. A solution would be to replace the matrix $R(\hat{n})$ by an instrumental variable matrix $Z(\hat{n})$ whose elements will not be correlated with the measurement noise.
Such an instrumental matrix $Z(\hat{n})$ can be obtained by replacing in the matrix $R(\hat{n})$, the columns $Y(t-1), Y(t-2), Y(t-3)$ by delayed version of $U(t-L-i)$, i.e., where $L > n$:

$$Z(\hat{n}) = [U(t-L-1), U(t-1), U(t-L-2), U(t-2) \cdots] \tag{2.95}$$

and therefore, the following criterion is used for the order estimation:

$$CV_{PJ}(\hat{n}, N) = \min_{\hat{\theta}} \frac{1}{N} \|Y(t) - Z(\hat{n})\hat{\theta}\|^2 + \frac{2\hat{n} log N}{N} \tag{2.96}$$

and:

$$\hat{n} = \min_{\hat{n}} CV_{PJ}(\hat{n}) \tag{2.97}$$

A typical curve of the evolution of the criterion (2.96) as a function of $\hat{n}$ is shown in Figure 8.5.2.

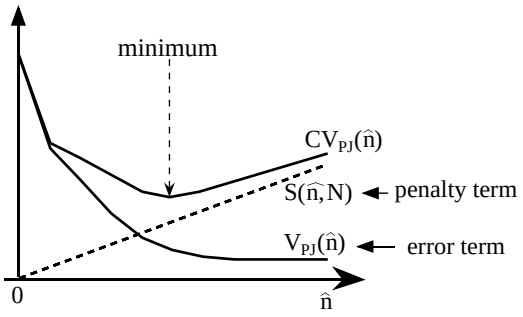


Figure 2.10: Evaluation of the criterion for order estimation

Once an estimated order $\hat{n}$ is selected, one can apply a similar procedure to estimate $\hat{n}_A, \hat{n} - \hat{d}, \hat{n}_B + \hat{d}$, from which $\hat{n}_A, \hat{n}_B$ and $\hat{d}$ are obtained.

2.6 Practical aspects of recursive closed loop identification

In practice, it is sometimes very difficult or even impossible to carry an open loop identification experiment (ex: integrator behaviour, open loop unstable). In some situation, a controller may already exist, therefore there would be no reason to open the loop if one needs to find a better model for improving the existing controller. A basic scheme for closed loop identification of a plant with a RST controller is presented in Figure 2.11.

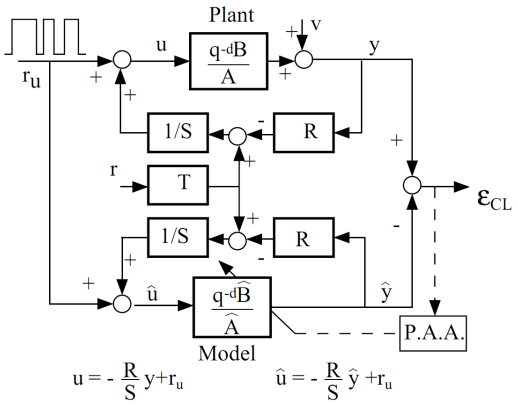


Figure 2.11: Closed loop identification scheme with PRBS added to the plant input

The objective is to estimate the parameters of the plant model defined by (2.57-2.59). The output of the plant oparating in closed loop is given by

$$y(t+1) = -A^* y(t) + B^* u(t-d) + Av(t+1) = \theta^T \varphi(t) + Av(t+1) \tag{2.98}$$

where $u(t)$ is the plant input, $y(t)$ is the plant output, $w(t)$ is the output disturbance noise and

$$\begin{aligned} \theta^T &= [a_1 \dots a_{n_A} b_1 \dots b_{n_B}] \\ \varphi^T(t) &= [-y(t) \dots - y(t-n_A+1) u(t-d) \dots u(t-n_B+1-d)] \\ u(t) &= -\frac{R}{S} y(t) + r_u(t) \end{aligned} \tag{2.99}$$

where $r_u(t)$ is the external excitation added to the control input.
Two more options are available for the position where the external excitation enters the closed loop system. In this case, in (2.99) $r_u(t)$ will be replaced by $r'_u(t)$:

- excitation added to the reference:
$$r'_u(t) = \frac{T}{S} r_u(t) \tag{2.100}$$
- excitation added to the measurement:
$$r'_u(t) = \frac{R}{S} r_u(t) \tag{2.101}$$

For a fixed value of the estimated parameters, the predictor of the closed loop is described by

$$\hat{y}(t+1) = -\hat{A}^* \hat{y}(t) + \hat{B}^* \hat{u}(t-d) = \hat{\theta}^T \phi(t) \tag{2.102}$$

where

$$\begin{aligned} \hat{\theta}^T &= [\hat{a}_1 \dots \hat{a}_{n_A} \hat{b}_1 \dots \hat{b}_{n_B}] \\ \phi^T(t) &= [-\hat{y}(t) \dots - \hat{y}(t-n_A+1) \hat{u}(t-d) \dots \hat{u}(t-n_B+1-d)] \\ \hat{u}(t) &= -\frac{R}{S} \hat{y}(t) + r_u(t) \end{aligned} \tag{2.103}$$

The closed loop prection (output) error is defined as

$$\epsilon_{CL}(t+1) = y(t+1) - \hat{y}(t+1) \tag{2.104}$$

The parameter adaptation algorithm remains the same as in the open loop recursive identification (2.51-2.54). Table 2.6 summarizes the characteristics of the algorithms used for recursive plant model identification in closed loop.

Table 2.6: Recursive closed loop identification algorithms

		Closed Loop Output Error (CLOE)	Filtered Closed Loop Output Error (FCLOE)	Extended Closed Loop Output Error (XCLOE)	Generalized Closed Loop Output Error (GCLOE)
Plant+Noise Model		$y = \frac{z^{-d}B}{A}u + v$		$y = \frac{z^{-d}B}{A}u + \frac{C}{A}e$	$y = \frac{z^{-d}B}{A}u + \frac{C}{AD}e$
Adjustable Parameter Vector		$\hat{\theta}^T(t) = [\hat{a}^T(t), \hat{b}^T(t)]$ $\hat{a}^T(t) = [\hat{a}_1 \dots \hat{a}_{n_A}]$ $\hat{b}^T(t) = [\hat{b}_1 \dots \hat{b}_{n_B}]$		$\hat{\theta}^T(t) = [\hat{a}^T(t), \hat{b}^T(t), \hat{c}^T(t)]$ $\hat{a}^T(t) = [\hat{a}_1 \dots \hat{a}_{n_A}]$ $\hat{b}^T(t) = [\hat{b}_1 \dots \hat{b}_{n_B}]$ $\hat{c}^T(t) = [\hat{c}_1 \dots \hat{c}_{n_C}]$	$\hat{\Theta}^T(t) = [\hat{a}^T(t), \hat{b}^T(t), \hat{h}^T(t)]$ $\hat{a}^T(t) = [\hat{a}_1 \dots \hat{a}_{n_A}]$ $\hat{b}^T(t) = [\hat{b}_1 \dots \hat{b}_{n_B}]$ $\hat{c}^T(t) = [\hat{c}_1 \dots \hat{c}_{n_C}]$ $\hat{d}^T(t) = [\hat{d}_1 \dots \hat{d}_{n_D}]$
Predictor Regressor Vector		$\varphi^T(t) = [-\hat{y}(t) \dots -\hat{y}(t-n_A+1),$ $\hat{u}(t-d) \dots \hat{u}(t-d-n_B+1)]$ $\hat{u}(t) = -\frac{R}{S}\hat{y}(t) + r_u$		$\varphi^T(t) = [-\hat{y}(t) \dots -\hat{y}(t-n_A+1),$ $\hat{u}(t-d) \dots \hat{u}(t-d-n_B+1)]$ $\epsilon_{CLf}(t) \dots \epsilon_{CLf}(t-n_H+1)]$ $\hat{u}(t) = -\frac{R}{S}\hat{y}(t) + r_u$ $\epsilon_{CLf} = \frac{1}{S}\epsilon_{CL}(t)$	$\varphi^T(t) = [-\hat{y}(t) \dots -\hat{y}(t-n_A+1),$ $\hat{u}(t-d) \dots \hat{u}(t-d-n_B+1)]$ $\epsilon_{CLf}(t) \dots \epsilon_{CLf}(t-n_H+1)$ $-\alpha(t) \dots -\alpha(t-n_D+1)]$ $\hat{u}(t) = -\frac{R}{S}\hat{y}(t) + r_u$ $\epsilon_{CLf} = \frac{1}{S}\epsilon_{CL}(t)$ $\alpha(t) = \hat{A}(t)\hat{y}(t) - \hat{B}(t)\hat{u}(t-d)$
Predictor Output	a priori	$\hat{y}^\circ(t+1) = \hat{\theta}^T(t)\varphi(t)$			
	a posteriori	$\hat{y}(t+1) = \hat{\theta}^T(t+1)\varphi(t)$			
Prediction Error	a priori	$\epsilon^\circ(t+1) = y(t+1) - \hat{y}^\circ(t+1)$			
	a posteriori	$\epsilon(t+1) = y(t+1) - \hat{y}(t+1)$			
Adaptation Error		$\nu^\circ(t+1) = \epsilon^\circ_{CL}(t+1)$			
Observation Vector		$\Phi(t) = \varphi(t)$	$\Phi(t) = \frac{S}{P}\varphi(t)$ $\hat{P} = \hat{A}S + z^{-d}\hat{B}R$	$\Phi(t) = \varphi(t)$	$\Phi(t) = \varphi(t)$

Chapter 3

How to use the application

The identification tab (Figure 3.1) appears after selecting the "Identification" button from the *Start* tab, Figure 1.1.
There are 2 possible modes for doing identification in iREG: automatic (Figure 3.1) and advanced (Figure 3.3).

3.1 Automatic use of the *Identification* tab

The automated identification tab is presented in Figure 3.3. The main objective is that of simplifying as much as possible the identification procedure, while maintaining good performance. For more advanced features see Section 3.2. The main features of the *Identification* tab are:

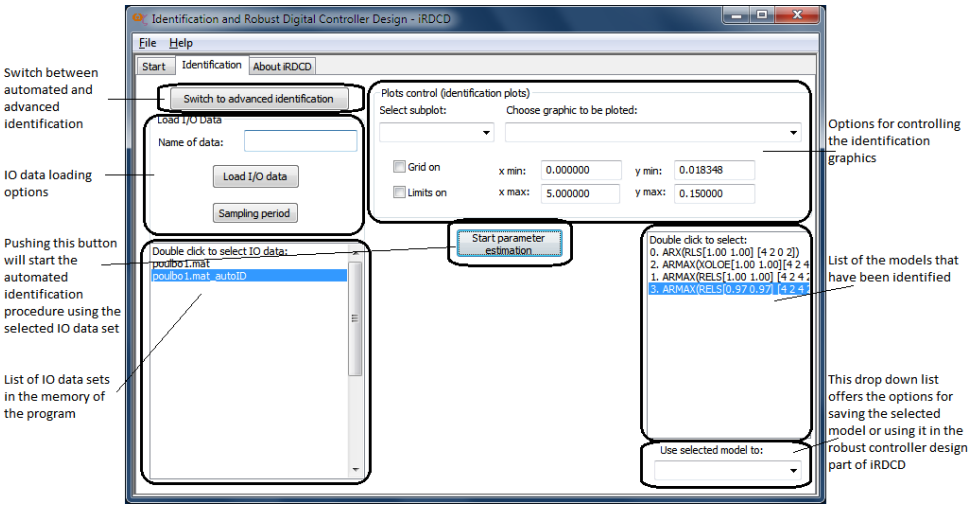


Figure 3.1: Tab for the automated mode of using the identification algorithms

- **switching between advanced and automated modes** - the tab will alternate view from Figure 3.1 to Figure 3.3 and back;
- **loading I/O data sets** - to types of input-output data sequences can be used in iREG: *basic text* files (**.dat*, **.txt* or **.c* extensions) and Matlab's **.mat* files. All input-output sequences loaded in iREG are shown in the *List of I/O data sets* (see lower left box in Figure 3.1). The next steps present the procedure for loading I/O data sequences in iREG:

  1. First, the user should introduce a short name for the I/O data sequence, that is going to be loaded in iREG, in the "Name of data:" text box (if this is left empty, not recommended, the name of the file on the disk will be used as name for the I/O data sequence).

2. If you are planing to load data from a text file, it is important to open it first in a text editor (e.g. Notepad) to see which column corresponds to the input and which to the output. Note that the accepted column separators are: space, tab, semicolon (;) and #.

   If you are loading data from a Matlab **.mat* file, make sure that it has been saved in the Matlab version 6 format (the proper command to use in Matlab is: *save -v6 file_name variables*).

3. Click on the "Load I/O data" button, select the appropriate type of file that you want to load, search for the file on the disk and than click Ok.

4. A new dialog box opens which is used for selecting the appropriate input-output data from the given file. For **.mat* files, one can select, by their names, the variables for the sampling time *Ts*, the *input* and the *output* sequences.

   For text files, one can select the appropriate column number for the input and the output sequences. Each drop down box gives the possibility to select a number from 1 to the total number of columns. If the total number of columns obtained in iREG is different from the expected one, after visual inspection of the text file, make sure that the right column separators are used, as indicated at step 2.

5. Finally, if it is necessary, the sampling period can be modified by clicking on the "Sampling period" button (see Figure 3.1).

- **selecting a different IO data set** from the list of all available ones;
- **modifying the parameters of the identification plots** - the user can select which graphs will appear and in what subplot (of the *Identification Plots* tab from the *Plots window*; each part of iREG, identification, robust digital controller design and controller order reduction, has its one tab(s) in the *Plots window*) and change the limits and grids of each graphic;
- **analyzing the identified models** - the user can plot different graphs in the subplots of the identification tab in the *Plots window*; among these, there are a few that can be used to analyze the model; these are strictly related to the selected model from the model list; therefore, the user can analyze different models be changing the selection from the model's list;
- **Start parameter estimation** - this will begin the identification procedure available in the automated mode; this is composed of:

  1. Filtering the data - the select IO data set will be filtered by removing the mean (centering the values on 0) and scaling (making the inputs match the outputs from the maximum value point of view);
  2. The new set of data is saved with the same name as the initial one but with an "_autoID" added to the end;
  3. A complexity estimation is run to find the orders of the model polynomials n_A, n_B and delay d
  4. Three open loop identification algorithms are used: *Recursive Least Squares (RLS)*, *Extended Recursive Least Squares (RELS)* and *Output Error with Extended Prediction Model (XOLOE)*; for all of them an *Decreasing gain* adaptation is used with an initial gain of 1000;
  5. A best model is identified by comparing the largest values of the prediction error autocorrelations;
  6. A new identification rutine is run, using the best model from the previous step with an Decreasint gain with variable forgetting factor adaptation algorithm ($\lambda_1 = 0.97$ and $\lambda_0 = 0.97$, see Section 2.3);
  7. Based on the same criterion as before, a new best model is selected from the two;
  8. If no valid model is found, a new identification is done with an increased order of the C polynomial ($n_C = max(n_A, n_B + d)$); only the *Extended Recursive Least Squares (RELS)* and *Output Error with Extended Prediction Model (XOLOE)* methods are used;
  9. In the end a check for common factors between polynomials A and B is done followed by a reidentification if necessary; furthermore, the program check for integrator or derivator in the identified model;
  10. All information regarding these steps is present in the final message from the program (e.g., Figure 3.2); the program gives the possibility to use the best model for designing a robust digital controller in the automatic mode as presented in Section 5.3.

- **saving the model or using it to compute a robust digital controller** - the model can be saved in WinPIM or Matlab version 6 formats; the model can also be send to the robust digital controller design part of iREG.

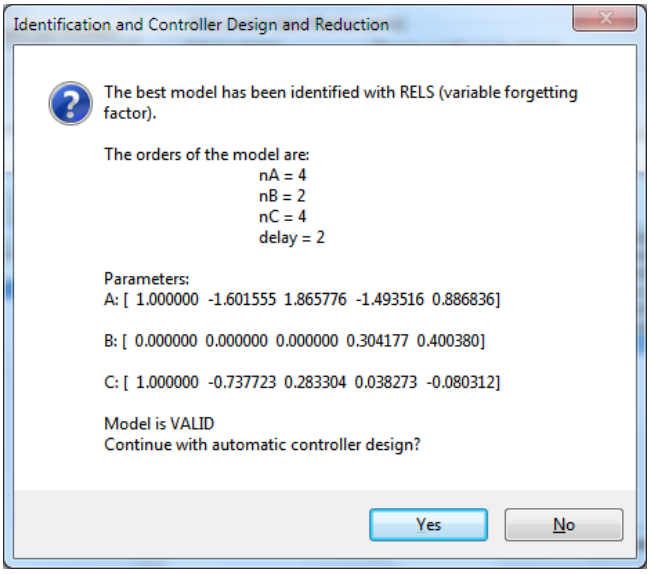


Figure 3.2: Example of message at the end of the automatic identification procedure

3.2 Advanced use of the *Identification* tab

In this section, the functionality of the advanced identification mode is presented. This tab is presented in Figure 3.3. There are 2 windows used for controlling the different parameters involved in the advanced identification

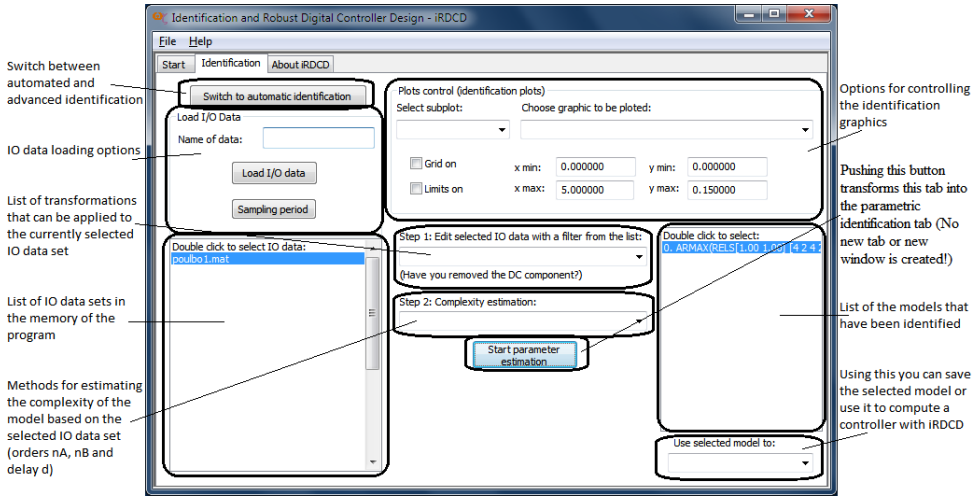


Figure 3.3: Tab for the advanced mode of using the identification algorithms

mode. The main window (Figure 3.3) offers control over:

- **switching between advanced and automated modes** - the tab will alternate view from Figure 3.3 to Figure 3.1 and back;
- **loading IO data sets** - to types of input-output data sequences can be used in iREG: *basic text* files (**.dat*, **.txt* or **.c* extensions) and Matlab's **.mat* files. All input-output sequences loaded in iREG are shown in the *List of I/O data sets* (see lower left box in Figure 3.1). The next steps present the procedure for loading I/O data sequences in iREG:

  1. First, the user should introduce a short name for the I/O data sequence, that is going to be loaded in iREG, in the "Name of data:" text box (if this is left empty, not recommended, the name of the file on the disk will be used as name for the I/O data sequence).
  2. If you are planing to load data from a text file, it is important to open it first in a text editor (e.g. Notepad) to see which column corresponds to the input and which to the output. Note that the accepted column separators are: space, tab, semicolon (;) and #.

     If you are loading data from a Matlab **.mat* file, make sure that it has been saved in the Matlab version 6 format (the proper command to use in Matlab is: *save -v6 file_name variables*).
  3. Click on the "Load I/O data" button, select the appropriate type of file that you want to load, search for the file on the disk and than click Ok.
  4. A new dialog box opens which is used for selecting the appropriate input-output data from the given file. For **.mat* files, one can select, by their names, the variables for the sampling time *Ts*, the *input* and the *output* sequences.

     For text files, one can select the appropriate column number for the input and the output sequences. Each drop down box gives the possibility to select a number from 1 to the total number of columns. If the total number of columns obtained in iREG is different from the expected one, after visual inspection of the text file, make sure that the right column separators are used, as indicated at step 2.
  5. Finally, if it is necessary, the sampling period can be modified by clicking on the "Sampling period" button (see Figure 3.1).

- **selecting a different IO data set** from the list that displays all the available data set; it should be pointed out that, by filtering the data a new data set is created that will be saved in this list, then the user can return to a previous data set by double clicking its name;
- **modifying the parameters of the identification plots** - the user can select which graphs will appear and in what subplot (of the *Identification Plots* tab from the *Plots window*; each part of iREG, identification, robust digital controller design and controller order reduction, has its one tab(s) in the *Plots window*) and change the limits and grids of each graphic;
- **IO data set filtering** - once there is at least one IO data set in the memory of the program , the user has the following options for modifying the selected data:
  - *Select range* - creates a new IO data set that contains only a part of the original one;
  - *Remove DC component* - creates a new IO data set that is centered on zero;
  - *Add offset* - creates a new IO data set that based on the original one by adding a scalar value to all it's components; the values added to the input and output can be different;
  - *Multiply by gain* - creates a new IO data set by multiplying all values of the original set with a scalar value; the value used for the inputs can be different from the value used for the outputs;
  - *Differentiate the output* - creates a new IO data set that has the same inputs but the outputs are filtered by $1 - z^{-1}$;
  - *Integrate the input* - creates a new IO data set that has the same outputs but the inputs are filtered by $\frac{1}{1 - z^{-1}}$;
  - *Differentiate the input* - creates a new IO data set that has the same outputs but the inputs are filtered by $1 - z^{-1}$;
  - *Apply digital filter* - creates a new IO data set by filtering with a user provided digital filter; the user can choose to filter only the inputs, only the outputs or both.

  the name field from the IO data loading group can also be used for naming the filtered data;
- **estimating the complexity of the process** - the two options available under this drop down list will return the values of N, n_A, n_B and delay d; these values should be known before starting any identification algorithm because they give the number of parameters that need to be estimated; the program will first give the estimated value of $N = max(n_A, n_B + d)$; the user can check this value and modify it if necessary; based on this, the other three will be calculated;
- **analyzing the identified models** - the user can plot different graphs in the subplots of the identification tab in the *Plots window*; among these, there are a few that can be used to analyse the model; these are strictly related to the selected model from the model list; therefore, the user can analyse different models be changing the selection from the model's list;
- **opening the parametric identification window** - this window (Section 3.2.1) gives access to all parameters involved in the identification procedures; the user can check and modify these values here; to estimate a model while in advanced mode you have to open this window;
- **the list of identified models** shows the last 30 models that have been identified in the current session of the program; the user can load a previous model by double clicking its name; a window appears asking if the current plots should be kept - for the purpose of comparing two or more models, one should chose to keep the current graphics and use appropriate subplots for plotting the desired graphics of the loaded model; another aspect of the model list is the name by witch a model is saved here - this information consists of the structure used for identification, the algorithm, values of λ_1 (forgetting factor) and λ_0 and the order of the polynomials of that model;
- **saving the model or using it to compute a robust digital controller** - the model can be saved in WinPIM or Matlab version 6 formats; the model can also be send to the robust digital controller design part of iREG.

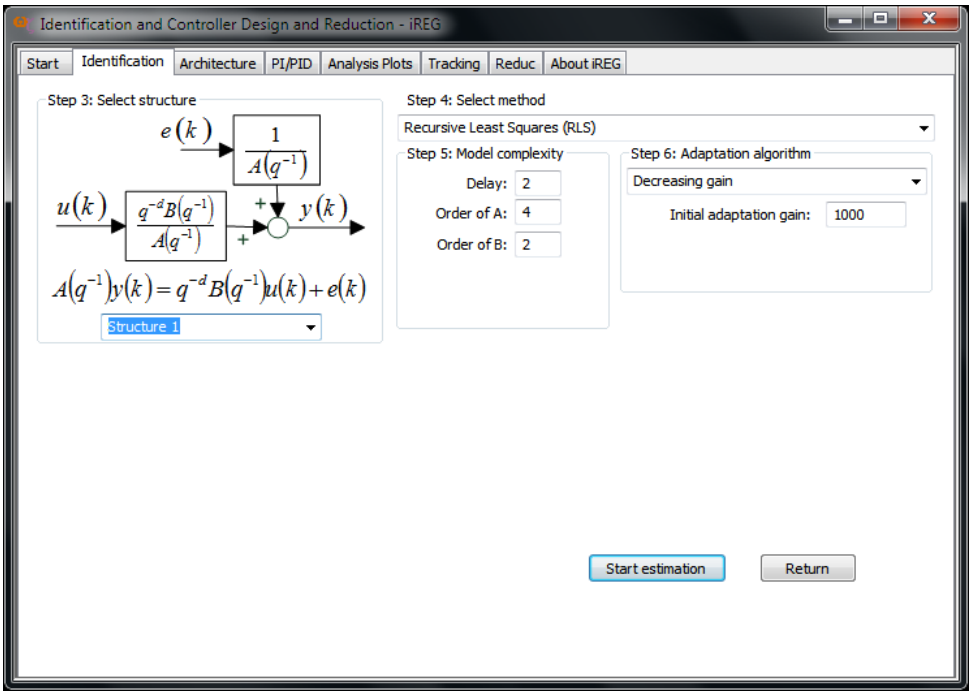


Figure 3.4: Parametric identification window - starting view

3.2.1 Parametric estimation window

A basic view of this window is shown in Figure 3.4. More options can be visible as provided by the chosen algorithm and structure (Figures 3.5 and 3.6).
To ease the use, this window has been divided into step, which should be executed in the given order. As can be observed, the first to steps are included in the main tab of identification (IO data set filtering and model order estimation). Once in the parametric estimation window, the user should select a structure for the model to be identified (Step 3):

- structures 1 - 4b are used for opened loop identification (the input signal to the process is only the frequency rich signal used to excite the process in the scope of finding a correct model on a large frequency spectrum, e.g., PRBS signal);
- structures 5 - 6b are used for closed loop identification (the input signal to the process is a combination of PRBS and command given by the controller); for closed loop identification, the user has to specify a controller (this can be done by choosing from a file - WinREG or Matlab version 6 format, by selecting the model that is currently loaded in the controller design part of iREG or by manually writing the coefficients of the R, S and T polynomials in the appropriate text boxes); also, the user should specify the position where the excitation signal enters the closed loop system - excitation added to the reference, controller output or measurement (see Section 2.6 for theoretical explanations);

Each structure has it's own set of algorithms from each the user should select in Step 4. The common set of parameters needed for each of the algorithms is given by the orders of the model's polynomials (Step 5 - n_A, n_B, delay d and for some other algorithms also n_C and n_D) and the adaptation algorithm's parameters (Step 6).
The adaptation algorithms are: "Decreasing gain", "Decreasing gain with constant forgetting factor" and "Decreasing gain with variable forgetting factor". Based on what is chosen, the user is given the option to modify the *Initial adaptation gain*, λ_1 (*Forgetting factor*) and λ_0 (*Lambda0*) - see Section 2.3 (*A.1*, *A.2* and *A.3*) for more information.
Some of the identification algorithms need that the user specifies some further parameters, presented next:

- *Output Error with Filtered Observations (FOLOE)* - needs a filter denominator to be specified; by default,

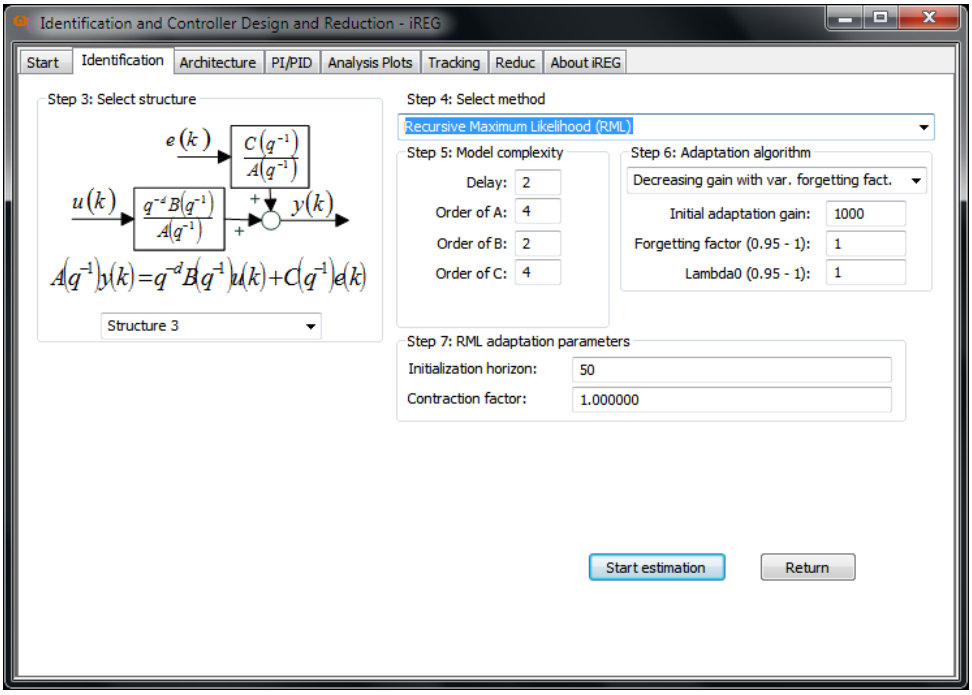


Figure 3.5: Parametric identification window - different view 1

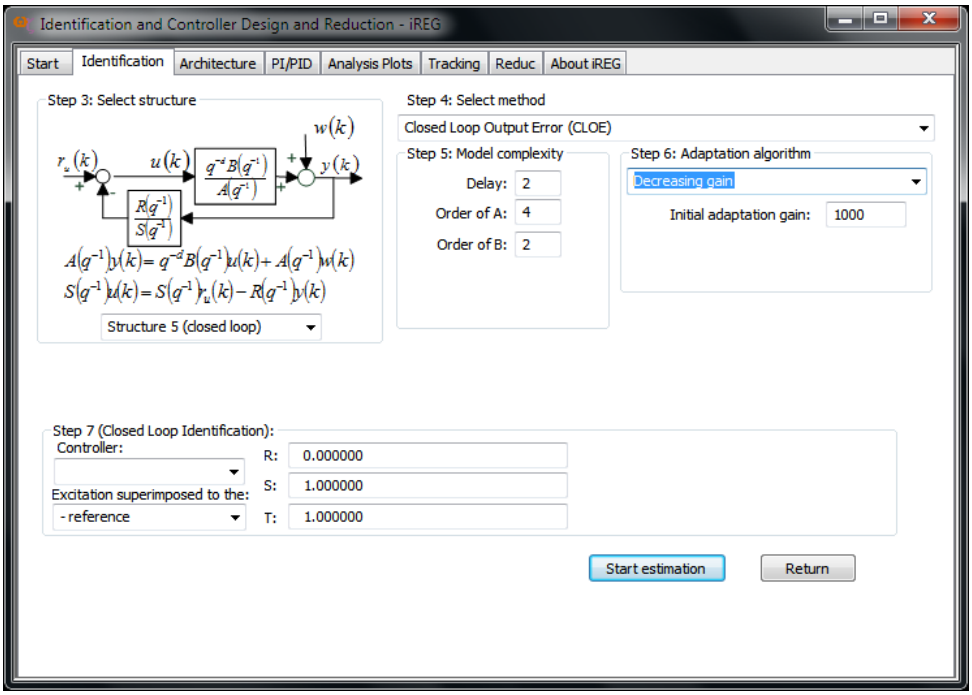


Figure 3.6: Parametric identification window - different view 2

this is the denominator of the last identified valid plant model; it can also be manually inputed by the user;

- *Recursive Maximum Likelihood (RML)* - the user has to specify some adaptation parameters (*Initial horizon* and *Contraction factor*); by default, these have the values 50 respectively 1.00;
- *Box Jenkins (BJ)* and *Generalized Closed Loop Output Error (GCLOE)* - are the only algorithms that use the order of a fourth polynomial D, n_D;
- *Closed Loop Output Error with Filtered Observations (FCLOE)* - an initial plant model is necessary; by default, this is the last valid identified model but can also be manually inputed, read from a file (WinPIM or Matlab version 6 format) or selected from the list of identified models;

Part II

Robust digital controller design

Chapter 4

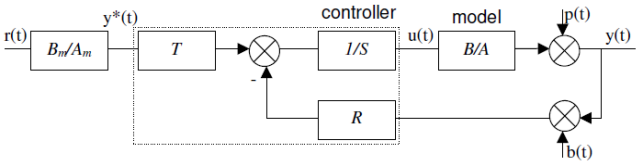
Basic theory

This chapter presents the key aspects of designing robust digital controllers.

4.1 Notation

A standard digital pole placement configuration using a polynomial controller (denoted R-S) is shown in Figure 4.1. The plant model $G(z^{-1})$ is of the form

$$G\left(z^{-1}\right) = z^{-d}\frac{B(z^{-1})}{A(z^{-1})} = z^{-d}\frac{b_1 z^{-1} + \ldots + b_{n_B} z^{-n_B}}{1 + a_1 z^{-1} + \ldots + a_{n_A} z^{-n_A}} \tag{4.1}$$



The R-S part of the controller has the transfer function:

$$\frac{R(z^{-1})}{S(z^{-1})} = \frac{R_0(z^{-1})H_R(z^{-1})}{S_0(z^{-1})H_S(z^{-1})} \tag{4.2}$$

where $H_R(z^{-1})$ and $H_S(z^{-1})$ denote the fixed parts of the controller (either imposed by the design or introduced in order to shape the sensitivity functions). $R_0(z^{-1})$ and $S_0(z^{-1})$ are solutions of the Bezout equation (poles of the closed loop):

$$A(z^{-1})S_0(z^{-1})H_S(z^{-1}) + z^{-d}B(z^{-1})R_0(z^{-1})H_R(z^{-1}) = \underbrace{P_D(z^{-1})P_F(z^{-1})}_{=P(z^{-1})} \tag{4.3}$$

where $P(z^{-1})$ represents the desired closed loop poles, $P_D(z^{-1})$ defines the dominant poles (specified) and $P_F(z^{-1})$ defines the auxiliary poles (which in part can be specified by design specifications and the remaining part is introduces in order to shape the sensitivity functions).
The tracking part $T(z^{-1})$ of the controller is used to compensate the closed loop dynamic in such way that the entire system transfer function (from $r(t)$ to $y(t)$) has the dynamic of the reference model $\frac{B_m(z^{-1})}{A_m(z^{-1})}$. The polynomial $T(z^{-1})$ is considered to have three basic forms:

- T contains all closed loop poles given by the polynomial $P = AS + BR$ and its static gain is adjusted so that the static gain of the transfer function from $y^*(t)$ to $y(t)$ is 1. Hence:

$$T(z^{-1}) = \frac{P(z^{-1})}{B(1)} \tag{4.4}$$

- T contains dominant closed-loop poles given by the polynomial P_D and its static gain is adjusted so the static gain of the transfer function from $y^*(t)$ to $y(t)$ is 1. Hence:

$$T(z^{-1}) = \frac{P_D(z^{-1})P_F(1)}{B(1)} \tag{4.5}$$

- T is a gain with the value:

$$T(z^{-1}) = \frac{P(1)}{B(1)} \tag{4.6}$$

The reference model $\frac{B_m(z^{-1})}{A_m(z^{-1})}$ is considered to be a second order transfer function with dynamics defined by natural frequency and damping. Sensitivity function shaping is the chosen method for assuring the desired controller and closed-loop performances. The considered sensitivity functions are:

- The output sensitivity function:

$$S_{yp}(z^{-1}) = \frac{A(z^{-1})S_0(z^{-1})H_S(z^{-1})}{P(z^{-1})} \tag{4.7}$$

- The input sensitivity function:

$$S_{up}(z^{-1}) = -\frac{A(z^{-1})R_0(z^{-1})H_R(z^{-1})}{P(z^{-1})} \tag{4.8}$$

- The complementary sensitivity function:

$$S_{yb}(z^{-1}) = -\frac{B(z^{-1})R_0(z^{-1})H_R(z^{-1})}{P(z^{-1})} \tag{4.9}$$

where S_{yp} is shaped to obtain a sufficient closed-loop robust stability, the shaping of S_{up} allows to limit controller gain and hence actuator effort and S_{yb} shaping help to limit noise sensitivity of the closed loop and it serves to fix a desired closed-loop tracking performance. More details can be found in [5, 1, 7].
We can now introduce the following parameterization:

$$\begin{array}{l} H_R(z^{-1}) = H'_R(z^{-1})\gamma_R(z^{-1}) \\ H_S(z^{-1}) = H'_S(z^{-1})\gamma_S(z^{-1}) \\ P_F(z^{-1}) = P'_F(z^{-1})\delta_R(z^{-1}) \\ P_F(z^{-1}) = P''_F(z^{-1})\delta_S(z^{-1}) \end{array} \tag{4.10}$$

With these notations we get:

$$\left|S_{yp}(z^{-1})\right| = \left|\frac{A(z^{-1})S_0(z^{-1})H'_S(z^{-1})}{P_D P''_F(z^{-1})}\right| \left|\frac{\gamma_S(z^{-1})}{\delta_S(z^{-1})}\right| \tag{4.11}$$

$$\left|S_{up}(z^{-1})\right| = \left|\frac{A(z^{-1})R_0(z^{-1})H'_R(z^{-1})}{P_D P'_F(z^{-1})}\right| \left|\frac{\gamma_R(z^{-1})}{\delta_R(z^{-1})}\right| \tag{4.12}$$

where the filters $F_{yp}(z^{-1}) = \frac{\gamma_S(z^{-1})}{\delta_S(z^{-1})}$ and $F_{up}(z^{-1}) = \frac{\gamma_R(z^{-1})}{\delta_R(z^{-1})}$ consist of several second order notch filters ($\frac{2zeros}{2poles}$ band-stop filters with limited attenuation) simultaneously tuned. The tuning means in fact searching for appropriate frequency characteristics of $\left|F_{yp}(z^{-1})\right|$ and $\left|F_{up}(z^{-1})\right|$. Specifically in our case, we are interested in frequency band-stop with limited attenuation characteristics and thus the tuning concerns the frequency of band-stop, its bandwidth and the maximum attenuation in the band-stop frequency.

4.2 Digital controller design procedure

Suppose that we dispose of the digital model G of the plant to be controlled. The controller design consists of the following steps:

1. *Design specifications* - Determine desired closed loop and tracking performances. The closed loop performances, such as robust stability, disturbance rejection, etc., has to be expressed by some templates imposed on sensitivity functions. The tracking properties include rise time, maximal overshoot, or settling time.
2. *Closed-loop part R-S design* - The sensitivity functions are shaped to satisfy design specifications (to enter the frequency responses to imposed templates). As we can see from the previous section, one disposes with the following design parameters:
   - P_D polynomial of desired dominant (the slowest) closed loop poles
   - $\frac{P'_F}{P''_F}$ polynomial of desired auxiliary closed loop poles
   - H_R fixed part of the controller numerator
   - H_S fixed part of the controller denominator
   - F_{yp} second order digital notch filters on S_{yp}
   - F_{up} second order digital notch filters on S_{up}

   which allows us to shape appropriately the sensitivity functions S_{yp}, S_{up} and S_{yb}.
3. *Tracking part design* - If the tracking properties are not satisfied by closed loop controller part R-S, the tracking part has to be designed. One has to choose an appropriate structure of T and to design the reference model $\frac{Bm}{Am}$ corresponding to the desired tracking performances. For reference model adjusting, the natural frequency and damping of the reference model denominator are modified.

4.3 Continuous PI/PID controller design

The following theory has been extracted from [9] and [8].

4.3.1 The KLV method for the omputation of PI controllers

1. Define G_0 the processes amplification at the zero frequency.
2. Define the frequency at which the total phase shit reaches -135^0 as ω_{135} and the amplification for which this phase shift is obtained as G_{135}.
3. Define the acceleration factor ($FactAcc$) as a value between 25% and 200%.
4. Compute the gain ratio $RapGain = \frac{G_{135}}{G_0}$ in the following manner:
   (a) $RapGain < 0.1 \Rightarrow \alpha = 1.15$
   (b) $0.1 \leq RapGain < 0.25 \Rightarrow \alpha = 1$
   (c) $0.25 \leq RapGain \Rightarrow \alpha = 1.15 \cdot RapGain + 0.75$
5. Compute $T_\Sigma = \frac{\alpha}{\omega_{135} \cdot FactAcc}$
6. Compute the parameters of the PI continuous controller ($H_{PI}(s) = K_p(1 + \frac{1}{K_i \cdot s})$):
   - If $RapGain < 0.25 \Rightarrow K_p = \frac{1}{2 \cdot \alpha \cdot \sqrt{1+\alpha^2} \cdot G_{135}}$; $K_i = 4 \cdot T_\Sigma$
   - If $RapGain \geq 0.25 \Rightarrow K_p = \frac{1}{2 \cdot G_0 \cdot (2.3 \cdot RapGain - 0.3)}$; $K_i = \frac{T_\Sigma}{2.3 \cdot RapGain - 0.3}$

4.3.2 The KLV method for the computation of PID controllers

1. Define G_0 the processes amplification at the zero frequency.
2. Define the frequency at which the total phase shit reaches -135^0 as ω_{135} and the amplification for which this phase shift is obtained as G_{135}.
3. Define the acceleration factor ($FactAcc$) as a value between 25% and 200%.
4. $T_{\Sigma_0} = \frac{1}{\omega_{135}}$
5. There are three ways to specify the filtered component of the derivative:
   (a) N imposed: $T_\Sigma = 1 + \frac{4}{N(4+FactAcc)} \cdot T_{\Sigma_0}$
   (b) $\frac{K_d}{N}$ imposed: $T_\Sigma = \frac{K_d}{N} + T_{\Sigma_0}$
   (c) $\frac{K_d}{N} = 0$: $T_\Sigma = T_{\Sigma_0}$
6. Compute the parameters of the PID continuous controller ($H_{PID}(s) = K_p(1 + \frac{1}{K_i \cdot s} + \frac{K_d \cdot s}{\frac{K_d}{N} s+1})$):
   - $K_p = \frac{4+FactAcc}{4} \cdot \frac{FactAcc}{2\sqrt{2} \cdot G_{135}}$
   - $K_i = \frac{4+FactAcc}{FactAcc} \cdot T_\Sigma$
   - $K_d = \frac{4}{4+FactAcc} \cdot T_\Sigma$

Chapter 5

How to use the application

This part of the program is composed of 7 tabs: *Architecture*, *Controller Design*, *Automated control design*, *PI/PID*, *Analysis Plots*, *Band-stop Filters* and *Tracking*. They will pe presented in the next sections.

5.1 Architecture tab

This is the main tab of Controller Design part of the software (Figure 5.1).

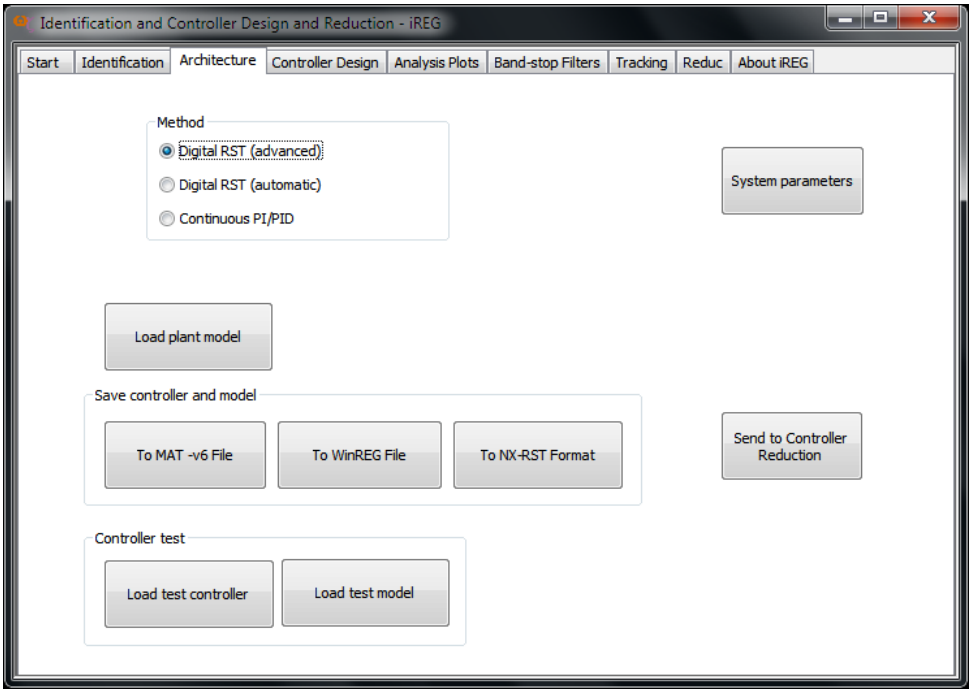


Figure 5.1: *Architecture* - main tab of controller design

This tab is divided in the following groups of buttons:

- "Method" - there are three approaches available for designing a controller. One can chose to design a digital RST controller and for this there are 2 available methods (advanced and automated) or chose to design a continuous PI/PID controller;
- "Load plant model" - one can use this button to load the polynomials A and B and the delay of a process that has been identified. The program reads Matlab (-v6) and WinPIM files;

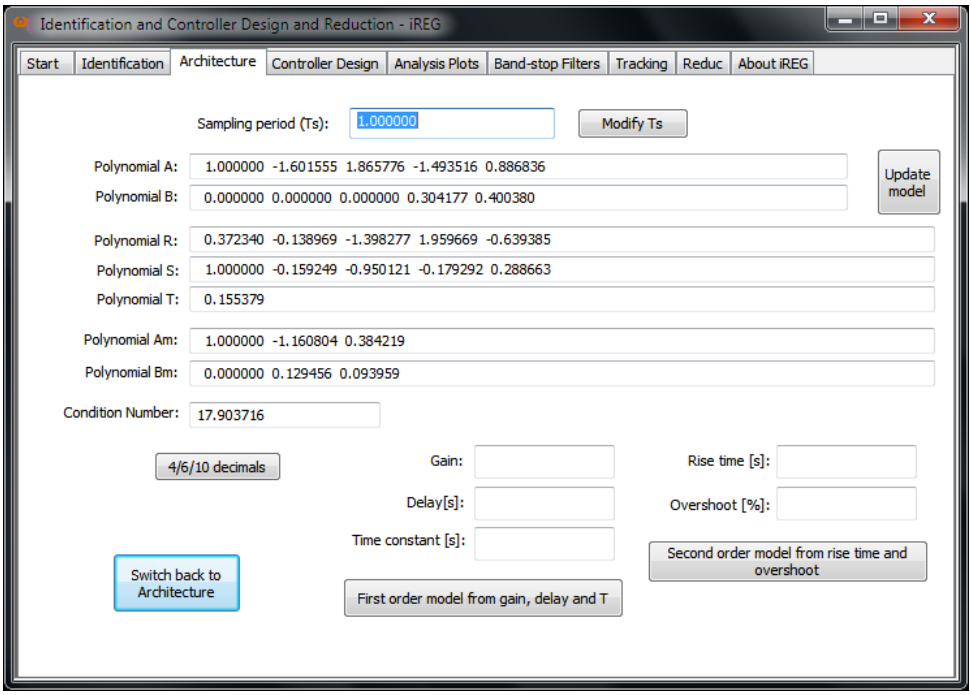


Figure 5.2: System parameters window

- "Save controller and model" - offers the possibility to save the designed digital controller to one of these file formats: Matlab (-v6), WinREG and NX-RST;
- "Load test controller" and "Load test model" - these group can be used to test a combination of model+controller. This offers the possibility to plot different graphs and to check the robustness indicators (modulus and delay margins);

There are another two buttons available:

- *System parameters*: switches the current "Architecture" tab to one that displays all current parameters from the controller design part of the program (Figure 5.2);
- *Send do Controller Reduction*: the controller that is currently being designed will be sent to controller reduction tool, together with the plant model that has been used here (Section 7). This allows for the design of a controller with a smaller number of parameters and a digital PID controller.

5.2 Advanced digital RST controller design

This tab can be viewed in Figure 5.3. It is very important to know that this tab is deactivated while using the "Digital RST (automatic)" or "Continuous PI/PID" methods and will not be visible.
There are three parts of this tab:

1. *Add Poles* is used to modify the closed loop poles and is situated in the upper part of the screen. In the right the poles of the closed loop system and of the plant are displayed. The program offers the possibility to add/remove complex/real closed loop poles:
   - to add a real multiple pole you have to click on the "New pole" button. Once you have added the new pole you can modify it to make it a complex pole by clicking on the radio buttons from the upper left part of the screen: "Complex conjugate poles", "Multiple pole";
   - one can also add a complex conjugate pole from the internal model of the plant by double clicking on the desired one from the list of plant poles.

The values of the poles can be modified by moving the cursors or by direct typing in the edit boxes. Furthermore the value of two complex conjugate poles can be calculated from the overshoot and rise time of a second order system. You just need to select a complex pole, write in the values for the overshoot M in percents and the rise time tr in seconds and click the *[nat.freq, damp]=fn(M,tr)* button.

To modify and/or delete an existing closed loop pole you first have to double click on it in the list. After that it will be selected and one can either modify its values or delete it.

The program offers the possibility to add as much fixed part to the S and R polynomials as needed. One just has to modify them in the corresponding sections.

2. *Hr(fixed part of R)* and *Hs(fixed part of S)* are used for modifying the fixed part of S and H polynomials.

   One can easily add/remove integrator or open the loop at 0.5 by clicking on the appropriate check box.

   The "Add cpl zero" and "Add real zero" buttons are used for adding as much fixed part to the controller as needed.

   As for the closed loop poles, there is a list of zeros for each polynomial. In this list one can easily see what has been added to the controller and can double click on one of them to modify its values.

   To delete an unnecessary zero one can click the "Delete" button after previously double clicking on it in the list.

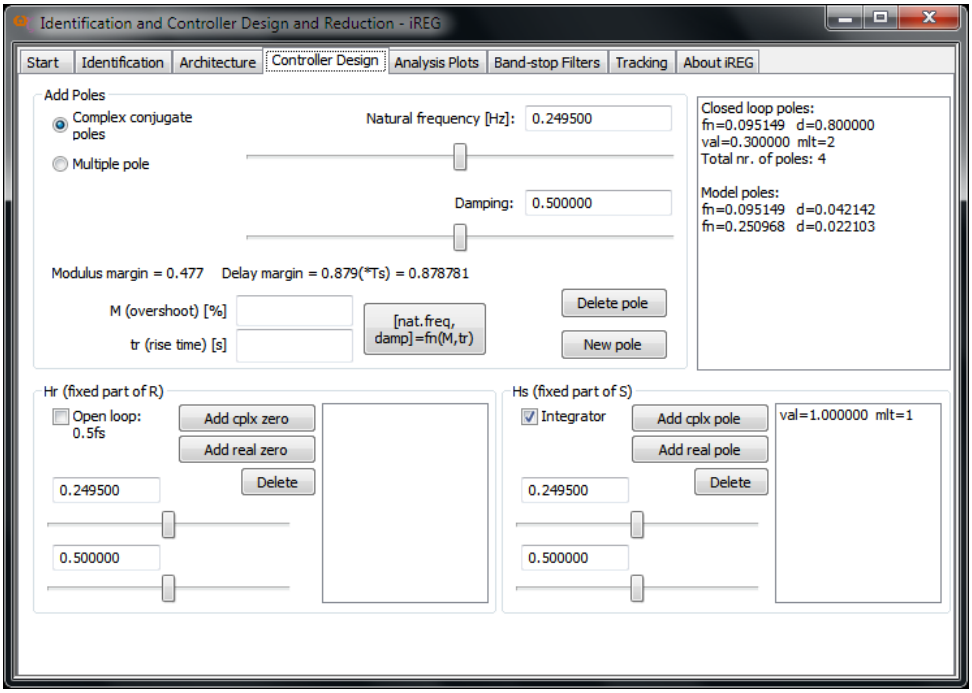


Figure 5.3: *Controller Design* - tab for advanced design of R and S polynomials

5.3 Automated digital RST controller design

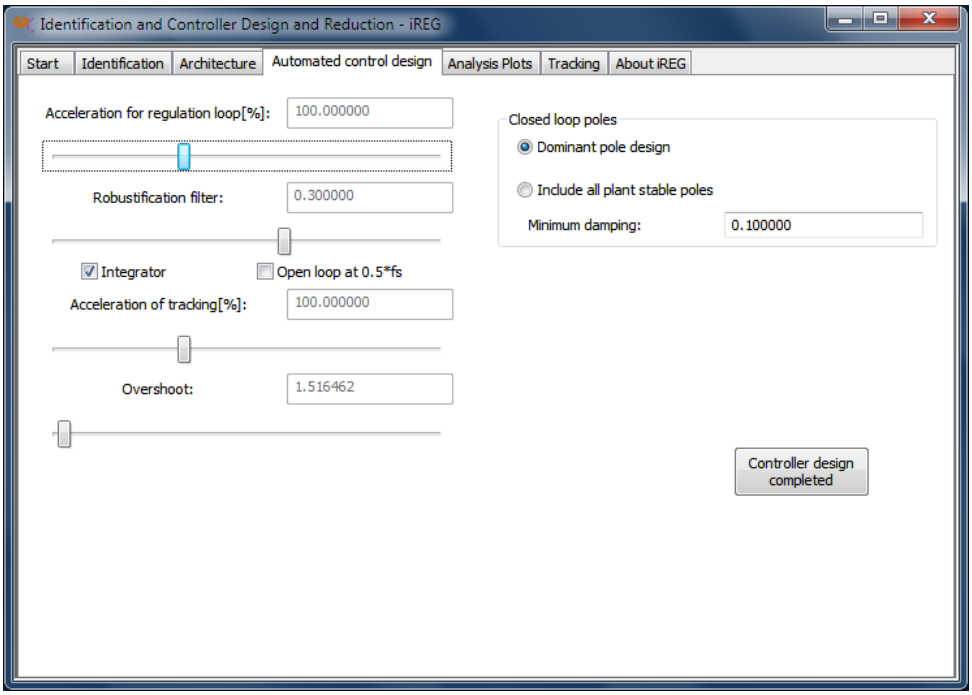


Figure 5.4: *Automated control design* - tab active only while in automated controller design mode. It facilitates the designing of the polynomials R and S

This tab (Figure 5.4) will only be active while using the "Digital RST (automatic)" method (selectable in the "Architecture" tab).
It serves for the easier designing of digital RST controllers but only offers a limited, although sufficient in most cases, number of parameters that can be tuned.
You can:

- modify the fixed part of the $S(q^{-1})$ polynomial by adding/removing the integrator;
- modify the fixed part of the $R(q^{-1})$ polynomial by choosing whether to open or not the loop at $0.5 \cdot f_S$;
- modify the characteristic polynomial $P(q^{-1})$ by changing the:
  1. regulation acceleration which is in fact the frequency of the dominant closed loop pole;
  2. robustification filter (this is the value of the real multiple closed loop pole);
  3. or chose to include or not all plant stable poles by selecting the appropriate radio button from the "Closed loop poles" group (he damping of these poles can be attenuated by a minimum value that can be specified here);
- modify the tracking model ($\frac{B_M(q^{-1})}{A_M(q^{-1})}$) by changing the:
  1. tracking acceleration which is the natural frequency for the tracking model;
  2. overshoot of the tracking model.

The button "Controller design completed" serves for easier utilization while using the automated mode of the program and switches to the "Architecture" tab where the user can save the controller or send it to the reduction part of the program.

5.4 Continuous PI/PID design using the KLV method

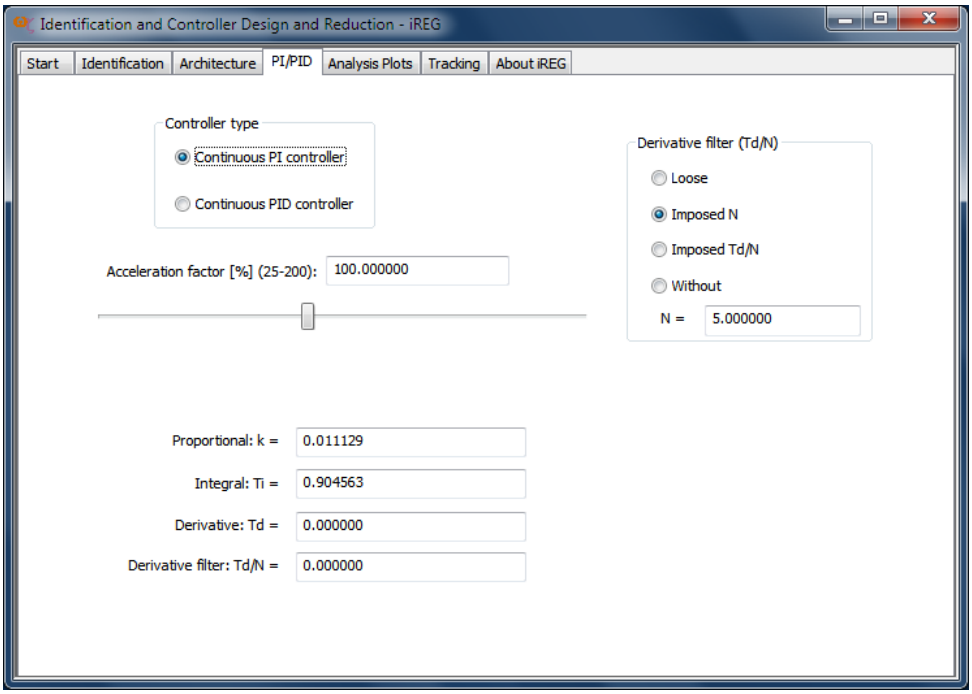


Figure 5.5: *PI/PID controller design* - tab active only while using the "Continuous PI/PID" method). It helps compute the parameters of a continuous PI or PID controller that is also discretized to the digital RST form so that it can be analysed

This tab (Figure 5.5) will only be active while using the "Continuous PI/PID" method (selectable in the "Architecture" tab). It can be used to compute the parameters of a continuous PI or PID controller. The program automatically computes the discretized $R(q^{-1})$ and $S(q^{-1})$ which it uses to compute the graphics. Also the program only saves the discretized R and S and the values of P, I, D and filter of D can be read in this tab.
In this tab, the user can:

- switch between PI and PID;
- change the acceleration factor for the KLV method (see 4.3 for more details);
- read the values of the parameters for the continuous regulator (P, I, D and the derivative of D);
- change the mode for computing the derivative's filter.

5.5 Analysis Plots

Fig.5.6 shows how this tab looks like. Using this, you can select the graphics that will be displayed on the "Analysis plots" tab of the Plot Window and modify their characteristics.

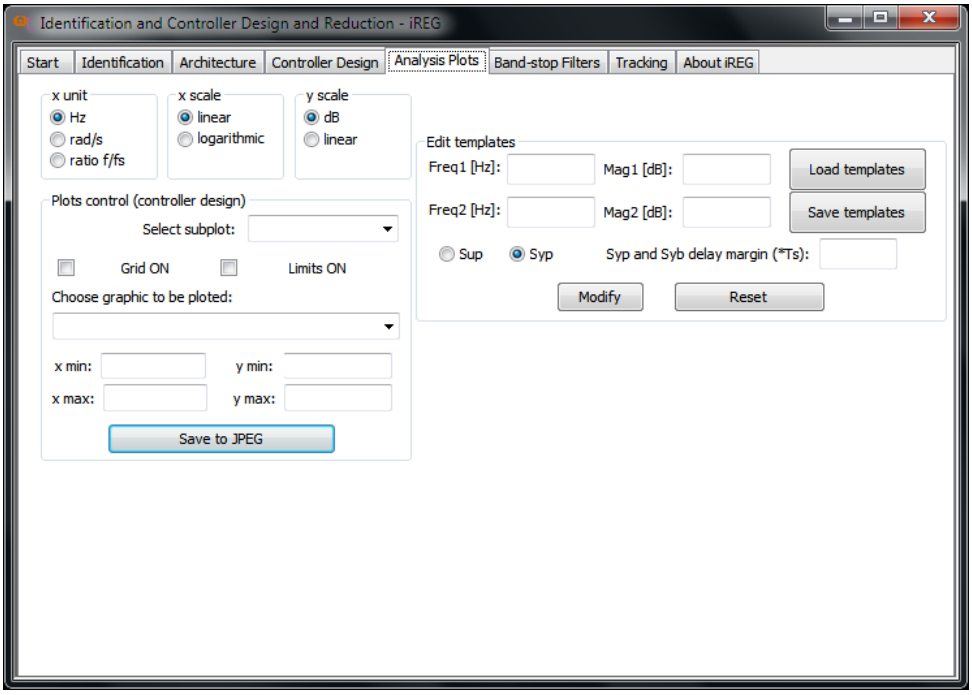


Figure 5.6: *Analysis Plots* - tab for controlling the graphics associated with controller design

This page also offers the possibility to modify the unit and scale of the x and y axis of the sensibility functions that are chosen for display. Note that the modifications apply to all sensibility function graphs.
The "Plots control (controller design)" group is used for selecting different graphics to be displayed on each one of the four subplots available in the *Analysis Plots*-tab of the *Plots window*. Furthermore, it is useful when one wants to show the grid or to modify the limits of the axes. Note that these settings can be made separately for each of the four graphs displayed by using the drop down list. One can also save as JPEG the selected graph by pressing the "Save to JPEG" button.
The "Edit templates" group can be used to modify the templates. There are two ways of modifying the templates. The first can be used independently for either Sup or Syp. It consists of choosing a starting frequency and a ending frequency together with a starting magnitude and a ending one. The template will then be modified by drawing a straight line between these two points. The second way to modify the templates is choosing a new delay margin. This will modify both Syp and Sup. The delay margin should be given as a multiple of the sampling period Ts. Once modified, the templates can be saved for latter use.

5.6 Band-stop Filters

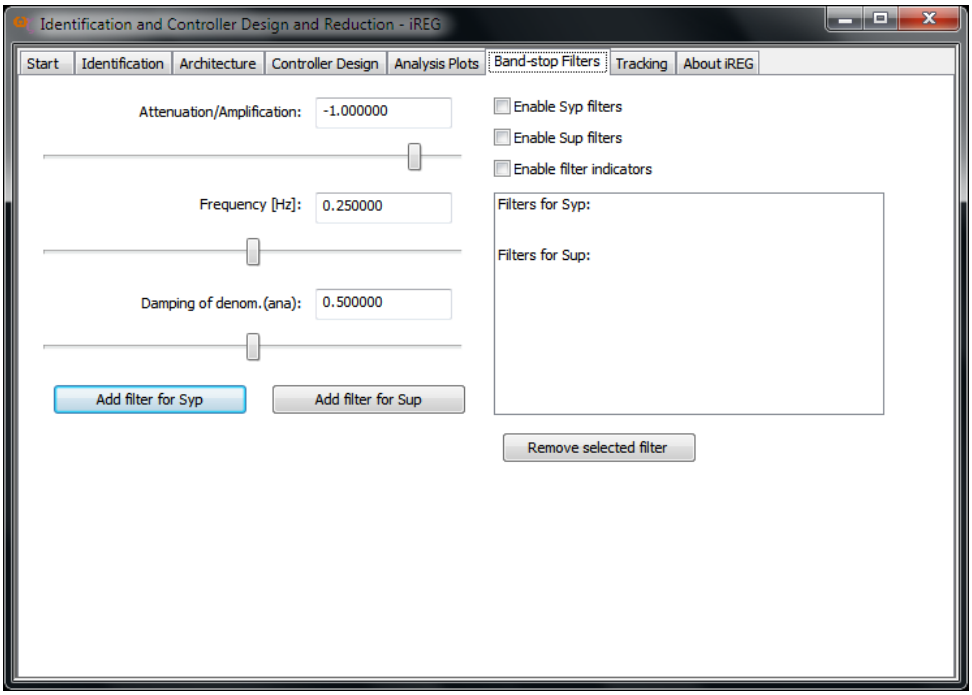


Figure 5.7: *Band-stop Filters* - tab used for designing second order notch filters

This window is used for the design of filters. It is possible to add filters for S_{yp} and S_{up}. Each filter has three parameters that can be adjusted:

- Attenuation/Amplification of the filter;
- Frequency at which the filter operates;
- Damping of denominator adjusts the width of the filter.

The list box on the right shows all filters that have been created. They are grouped by the sensitivity function that they influence directly (either S_{yp} or S_{up}). The filters on either one of the sensitivity functions can be enabled or disabled by clicking one of the appropriate check boxes.
The "Enable filter indicators" will put lines on the graphs of S_{yp} and S_{up} at the frequencies of the filters that are enabled as in Figure 5.8.
To modify one filter first you have to double click on it in the list of filters. Afterwards, it can be modified or removed.

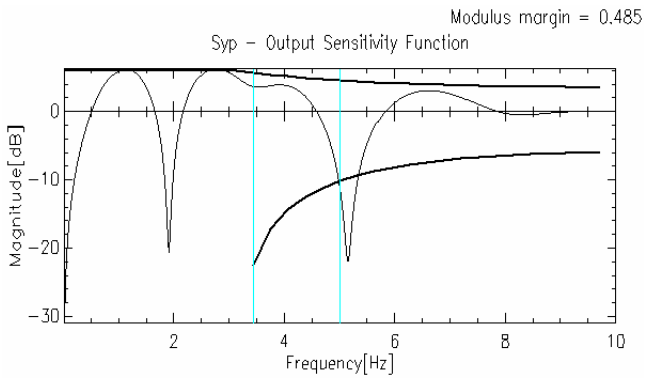


Figure 5.8: *Output sensitivity function with filters and indicators*

5.7 Tracking

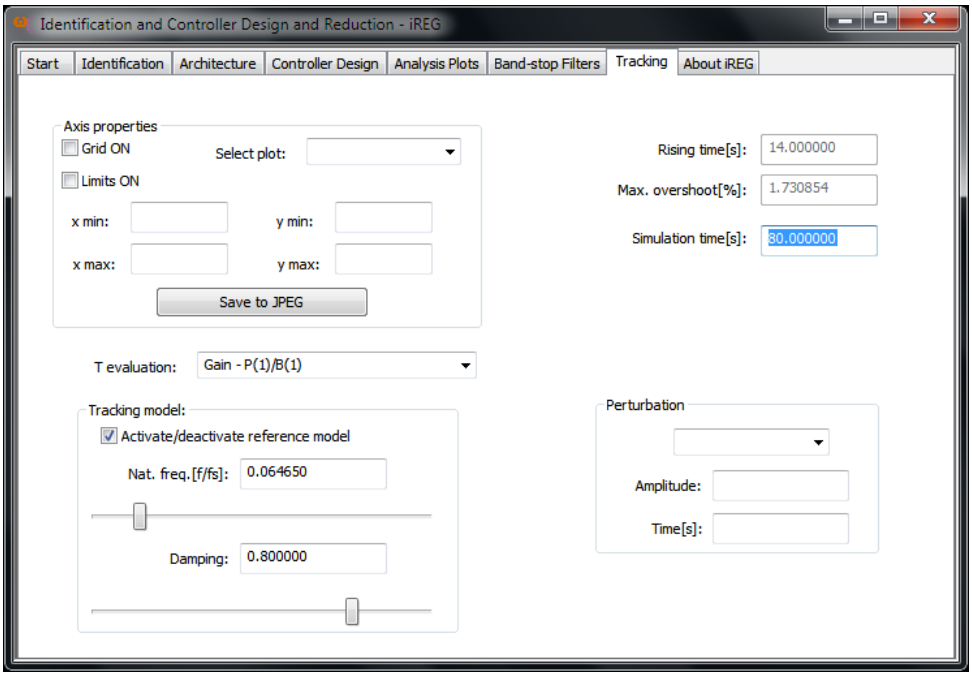


Figure 5.9: *Tracking* - tab for computing the tracking part of a controller (reference model and polynomial T)

This tab (Figure 5.9) is used for designing the tracking part of the controller (polynomial T) and the reference model ($\frac{B_m}{A_m}$).
For the T polynomial there are three possible calculation methods that can be chosen from a dropdown list called "T evaluation" (see reference [5] for more information):

- *All CL poles - Pd*Pf/B*(1)* - $T(z^{-1})$ is a polynomial with adjusted static gain containing all closed loop poles;
- *Dominant CL poles Pd*Pf(1)/ B*(1)* - $T(z^{-1})$ is a polynomial with adjusted static gain containing one complex pair of roots corresponding to closed loop dominant poles defined by P_d;
- *Gain P(1)/B(1)* - $T(z^{-1})$ is a constant with an appropriate value so that the static gain of the closed loop system is equal to one.

The reference model can be enabled or disabled. By disabling it, the reference's model transfer function becomes equal to 1 ($B_m(z^{-1}) = A_m(z^{-1}) = 1$). If enabled, it's transfer function is obtained by discretization of a continuous second order transfer function with normalized natural frequency and damping given by the user in the "Tracking model" group of this tab.
Note that in the automated controller design mode, one can only enable/disable the reference model but it's dynamics cannot be modified! The polynomial $T(z^{-1})$ can be changed as in advanced controller design mode.
"Rising time" gives the system rising time (time to reach 90% of the final stationary value) in seconds. "Max. overshoot" represents the relative difference between the maximum value and the final stationary value in percents (%). "Simulation time" shows the total simulation time. This value can be edited and changed.
"Perturbation" group can be used to add a step perturbation to the output of the system. "Axis properties" group can be used in a way similar to the "Plots control" group of the "Analysis plots" tab but with effect on the tracking plots.
All the tracking plots are displayed in the third tab of the Plots Window called "Tracking Plots". An example of this window can be seen in Figure 5.10. You can see in this figure how the perturbation is rejected with the current controller.

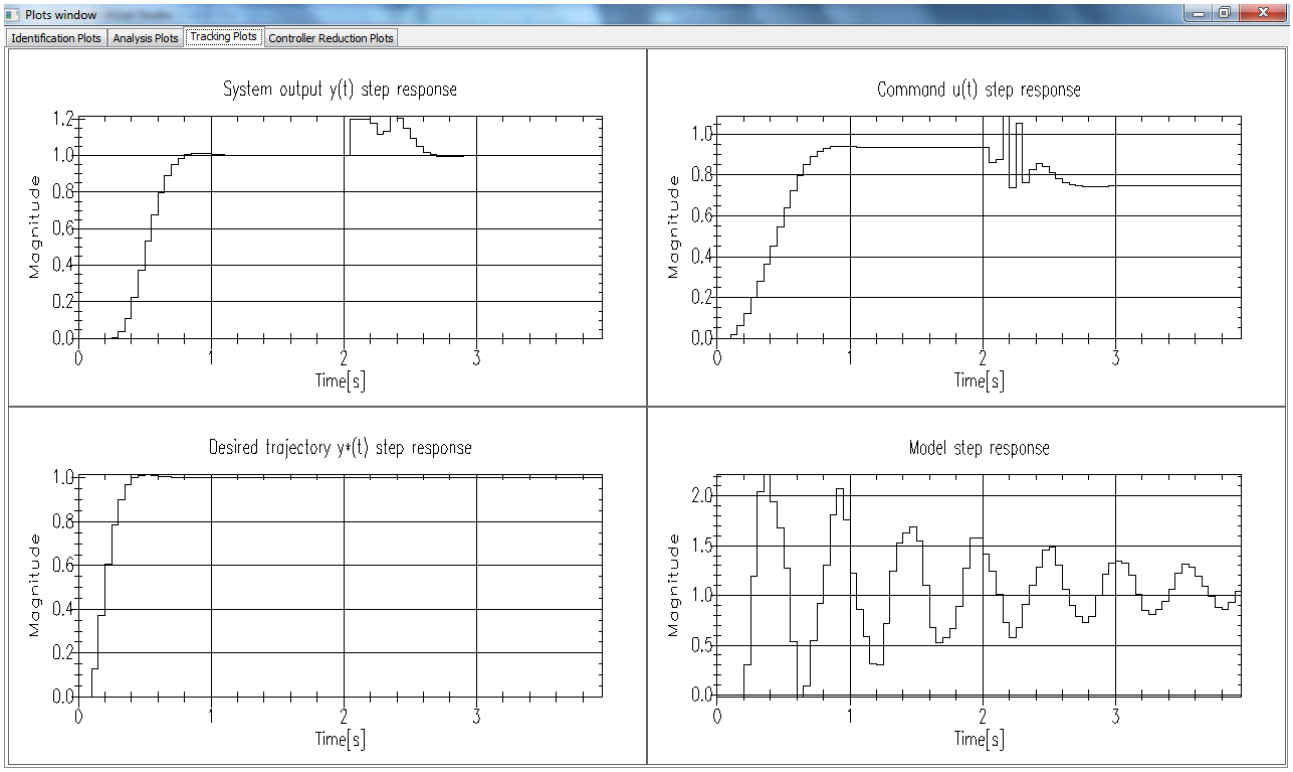


Figure 5.10: Tracking Plots tab of the Plots window

Part III

Controller reduction

Chapter 6

Basic theory

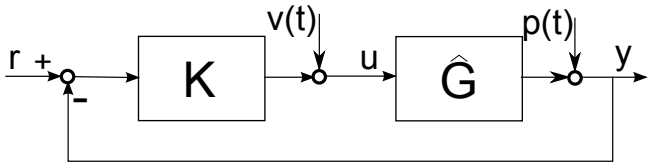


Figure 6.1: Nominal closed loop system

This chapter addresses the problem of directly estimating the parameters of a reduced order digital controller using a closed loop type identification algorithm. The algorithm minimizes the closed loop plant input error between the nominal closed loop system and the closed loop system using the reduced order controller ([3] and [2]). It is assumed that a plant model (if necessary validated in closed loop with the nominal controller) is available.
In the given nominal system from Figure 6.1, the $\hat{G}$ is the identified plant model

$$\hat{G}\left(z^{-1}\right) = \frac{z^{-d}\hat{B}\left(z^{-1}\right)}{\hat{A}\left(z^{-1}\right)}$$

where

$$\begin{aligned} \hat{B}(z^{-1}) &= \hat{b}_1 z^{-1} + \cdots + \hat{b}_{n_B} z^{-n_B} = z^{-1}\hat{B}^*(z^{-1}) \\ \hat{A}(z^{-1}) &= 1 + \hat{a}_1 z^{-1} + \cdots + \hat{a}_{n_A} z^{-n_A} \\ &= 1 + z^{-1}\hat{A}^*(z^{-1}) \end{aligned} \tag{6.1}$$

and K is the nominal controller

$$K\left(z^{-1}\right) = \frac{R\left(z^{-1}\right)}{S\left(z^{-1}\right)}$$

where

$$\begin{aligned} R(z^{-1}) &= r_0 + r_1 z^{-1} + \cdots + r_{n_R} z^{-n_R} \\ S(z^{-1}) &= 1 + s_1 z^{-1} + \cdots + s_{n_S} z^{-n_S} \\ &= 1 + z^{-1}S^*(z^{-1}) \end{aligned} \tag{6.2}$$

The following sensitivity funtions will be usefull in this presentation:

- output sensitivity function

$$S_{yp}(z^{-1}) = \frac{1}{1+KG} = \frac{A(z^{-1})S(z^{-1})}{P(z^{-1})}$$

- input sensitivity function

$$S_{up}(z^{-1}) = -\frac{K}{1+KG} = -\frac{A(z^{-1})R(z^{-1})}{P(z^{-1})}$$

- output sensitivity function with respect to an input disturbance

$$S_{yv}(z^{-1}) = \frac{G}{1+KG} = \frac{z^{-d}B(z^{-1})S(z^{-1})}{P(z^{-1})}$$

- complementary sensitivity function

$$S_{yr}(z^{-1}) = \frac{KG}{1+KG} = \frac{z^{-d}B(z^{-1})R(z^{-1})}{P(z^{-1})}$$

where

$$P(z^{-1}) = A(z^{-1})S(z^{-1}) + z^{-d}B(z^{-1})R(z^{-1})$$

The algorithms that will be used to find the reduced order controller are called: *Closed Loop Input Matching (CLIM)with excitation added to the controller input* (Figure 6.2) and *Closed Loop Input Matching (CLIM)with excitation added to the plant input* (Figure 6.3). These are very similar to the *Closed Loop Output Error (CLOE)* algorithms used for recursive plant model identification in closed loop.

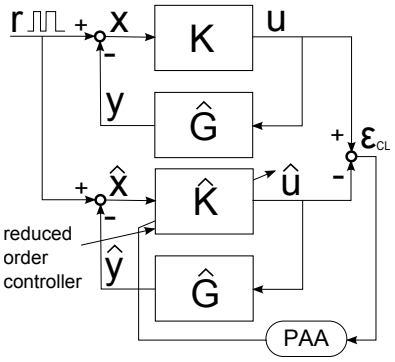


Figure 6.2: Closed Loop Input Matching with excitation added to the controller input

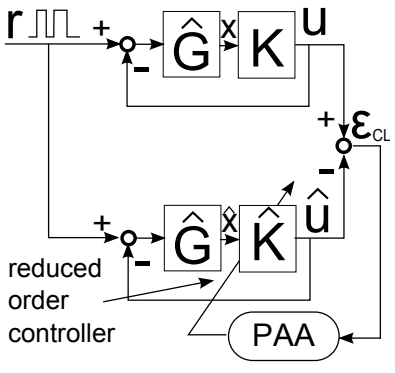


Figure 6.3: Closed Loop Input Matching with excitation added to the plant input

The signal $x(t)$ is defined as: $x(t) = r(t) - y(t)$ in Figure 6.2 and $x(t) = \hat{G}(r(t) - u(t))$ in Figure 6.3. Using this, we get the *a priori* input:

$$\hat{u}^o(t+1) = -\hat{S}^*(t, z^{-1})\hat{u}(t) + \hat{R}(t, z^{-1})\hat{x}(t) = \hat{\theta}_c^T(t)\phi_c(t)$$

and the *a posteriori* input:

$$\hat{u}(t+1) = \hat{\theta}_c^T(t+1)\phi_c(t)$$

where for the scheme of Figure 6.2

$$\hat{x}(t+1) = -\hat{A}^*(z^{-1})\hat{x}(t) - \hat{B}^*(z^{-1})\hat{u}(t-d) + \hat{A}(z^{-1})r(t)$$

and for the scheme of Figure 6.3, we have

$$\hat{x}(t+1) = -\hat{A}^*(z^{-1})\hat{x}(t) - \hat{B}^*(z^{-1})\hat{u}(t-d) + \hat{B}^*(z^{-1})r(t-d)$$

The unknowns (controller parameters) are

$$\hat{\theta}_c^T(t) = [\hat{s}_1(t), \ldots, \hat{s}_{n_S}(t), \hat{r}_0(t), \ldots, \hat{r}_{n_R}(t)]$$

and the measurements (available values)

$$\phi_c^T(t) = [-\hat{u}(t), \ldots, -\hat{u}(t-n_S+1), \hat{x}(t+1), \ldots, \hat{x}(t-n_R+1)]$$

The *a priori* closed loop input error will be given by

$$\hat{\epsilon}_{CL}^o(t+1) = u(t+1) - \hat{u}^o(t+1)$$

and the *a posteriori* by

$$\hat{\epsilon}_{CL}(t+1) = u(t+1) - \hat{u}(t+1)$$

For each of the algorithms (excitation added to the controller/plant input), another filtered version, *FCLIM*, can be considered by taking

$$\phi(t) = \frac{\hat{A}(z^{-1})}{\hat{P}(z^{-1})}\phi_c(t) \tag{6.3}$$

while in the case of the nonfiltered algorithm, *CLIM*,

$$\phi(t) = \phi_c(t) \tag{6.4}$$

The *parameter adaptation algorithm (PAA)* is described by these equations:

$$\hat{\theta}_c^T(t+1) = \hat{\theta}_c^T(t) + F(t)\phi^T(t)\hat{\epsilon}_{CL}(t+1) \tag{6.5}$$

$$F^{-1}(t+1) = \lambda_1(t)F^{-1}(t) + \lambda_2(t)\phi^T(t)\phi^T(t) \tag{6.6}$$

$$0 < \lambda_1(t) \leq 1; 0 \leq \lambda_2(t) \leq 2; F(0) > 0$$

$$\hat{\epsilon}_{CL}(t+1) = \frac{\hat{\epsilon}_{CL}^o(t+1)}{1 + \phi^T(t)F(t)\phi(t)} \tag{6.7}$$

Chapter 7

How to use the application

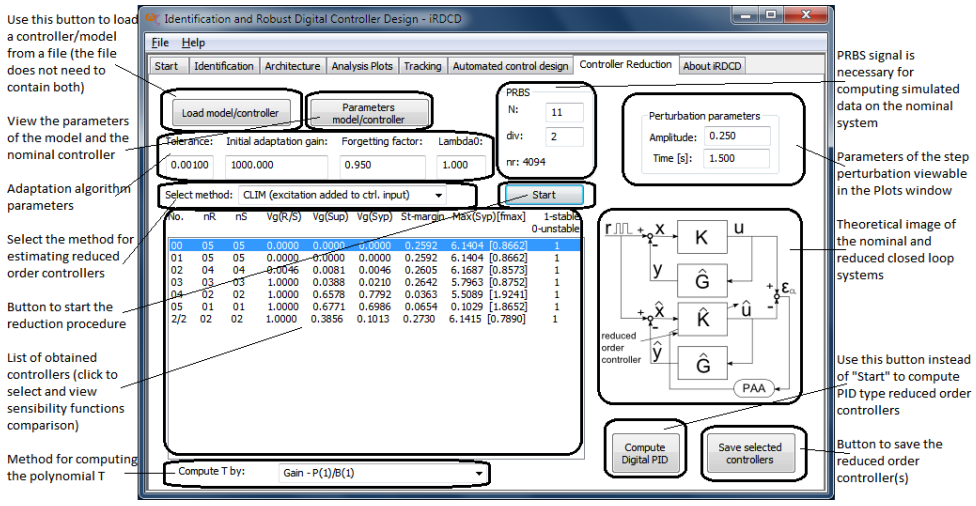


Figure 7.1: Controller reduction tab

Figure 7.1 gives an explained view over the controller reduction tab of iREG. The different parts of this tab will be described next:

- *Load model/controller* - use this to load a model and/or a controller, if not imported directly from the controller design part of iREG (Section 5.1);
- *Parameters model/controller* - will open a window like the one in Figure 7.2
- Adaptation algorithm parameters - *initial adaptation gain (GI)* as described by (2.73), *forgetting factor* (λ_1), λ_0 as in Section 2.3 and the *tolerance* for computing the VGap distance of two sensitivity functions;
- Method selection drop down list - four method are available based on the CLIM algorithm, with excitations added to the plant or controller input and also with or without filtered observation (Section 6);
- Parameters of the PRBS signal - frequency divider and number of bits in the shift register; the program calculates the period of the signal (*nr. 4094* in the given screen shot);
- *Start* button - will begin the identification of reduced order controller using the parameters provided by the user for the nominal system, *tolerance*, *initial adaptation gain*, *forgetting factor*, $lambda_0$ and PRBS signal generator and the choosen method;
- List of obtained controllers - the first in the list is the nominal controller, the others are the ones computed by the program; the entries in the list that are marked with blue color, will be shown in the *Plots window*,

Controller Reduction Plots tab; the nominal system's graphics are always shows so that the user can compare this with different reduced order controller

- *Compute T by* - two possibilities are provided for calculating the polynomial T of the RST controller (for the case of digital PID controller, the T will be left unchanged $T(q^{-1}) = R(q^{-1})$):
  - *All CL poles - P/B*(1)* - $T(z^{-1})$ is a polynomial with adjusted static gain containing all closed loop poles;
  - *Gain P(1)/B(1)* - $T(z^{-1})$ is a constant with an appropriate value so that the static gain of the closed loop system is equal to one;
- *Perturbation parameters* - the user has the possibility to check the reduced order controller on a step reference (magnitude 1) and with a step disturbance; the time of appearence and amplitude of the perturbation can be chosen by the user; the effect on the closed loop system with nominal/reduced controller can be viewed in the *Plots window*, *Controller Reduction Plots* tab;
- Theoretical image of the two systems with nominal and reduced order controllers; this image is in accordance with the algorithm used for estimation;
- *Compute Digital PID* - this button can be used as an alternative to *Start*; it will compute two digital PID controllers using the filtered versions of the CLIM algorithms (excitation on controller respectively plant input); the controllers are of second order with integrator; the $T(q^{-1})$ polynomial is equal to $R(q^{-1})$ and the reference modes is absent ($\frac{B_M(q^{-1})}{A_M(q^{-1})} = 1$) so that the graphics will show the actual performances if implemented on a PID control structure;
- *Save selected controllers* - this will save the selected controller(s) to a single Matlab version 6 file (in the future, there will be also the possibility to save in NX-RST format).

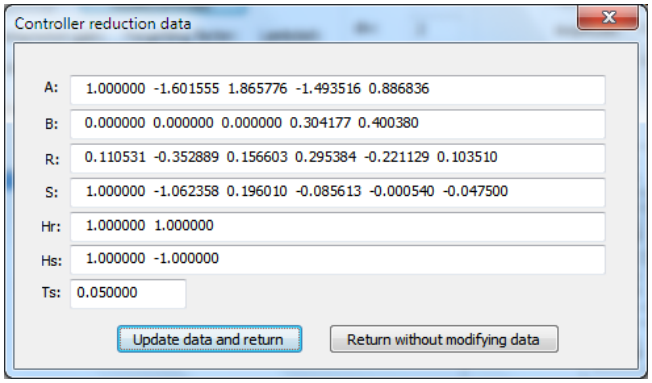


Figure 7.2: Model and controller parameters that are currently used to describe the nominal system

It is recomended to chose:

- Tolerance: 0.001;
- Initial adaptation gain: 1000;
- Forgetting factor (λ_1): 0.95-1;
- λ_0: 0.5-1;
- PRBS frequency divider, div: 1 or 2;
- PRBS length of shift register: 11.

Bibliography

[1] I. D. Landau and A. Karimi. Robust digital control using pole placement with sensitivity function shaping method. *Int. J. Robust and Nonlin. Cont.*, 1998.

[2] I. D. Landau and A. Karimi. A unified approach to model estimation and controller reduction (duality and coherence). *European Journal of Control*, 2002.

[3] I. D. Landau, A. Karimi, and A. Constantinescu. Direct controller order reduction by identification in closed loop. *Automatica*, 37:1689–1702, 2001.

[4] I. D. Landau, R. Lozano, M. M'Saad, and A. Karimi. *Adaptive control*. Springer, London, 2nd edition, 2011.

[5] I. D. Landau and G. Zito. *Digital Control Systems - Design, Identification and Implementation*. Springer, London, 2005.

[6] L. Ljung and T. Söderström. *Theory and practice of recursive identification*. The M.I.T Press, Cambridge Massachusetts , London, England, 1983.

[7] Hynek Prochzka and Ioan Dor Landau. Pole placement with sensitivity function shaping using 2nd order digital notch filters. *Automatica*, 39(6):1103 – 1107, 2003.

[8] A. Voda and I. D. Landau. The autocalibration of pi controllers based on two freqeuncy measurements. *International Journal of Adaptive Control and Signal Processing*, 9:395–421, 1995.

[9] A. Voda and I. D. Landau. A method for the auto-calibration of pid controllers. *Automatica*, 31:41–53, 1995.